

SECTION 1 Introduction to Programming

General Information

The Series **One™** and **Series One™** Plus are easy to use small PCs with many advanced features. One of the advantages of these PCs is the ability to be programmed or tailored specifically to the needs of an application. Furthermore, if requirements change, the PC can be reprogrammed to fit the new application. This chapter provides the information a user needs to develop or modify the logic within the Series One or Series One Plus. It will discuss the basics of CPU operation necessary to develop a proper logic program, the theory behind each function, examples of how that function could be used, and step-by-step **entry** of sample programs.

Planning a PC System

When planning a **PC** system, the first step should be to **define** the system by writing a description of the **functional** requirements for that system. A description would usually consist of block diagrams and written descriptions of the various parts of the system. Input **devices** should be defined along with the process or machines to be **controlled**. The next step would be to develop the program required to control the system.

On a sheet of paper, **layout each rung of your ladder diagram and assign references to** inputs and outputs. The program can also be **written in mnemonic form (Boolean), which is how** it will be keyed into the Series One or Series One Plus when using the hand-held programmer. As an option, programs can be entered in ladder diagram Format using the Portable Programmer. This applies to both the Series One and Series One Plus. You can also enter programs with the **Workmaster®** industrial computer using **Logicmaster™1** application software. For **information** on using the **Workmaster** industrial computer for **programming**, refer to the **Logicmaster 1 Programming** manual, **GEK-96632** for Series One. For Series One Model E, Series One Plus, and Series One Plus **3.7K**, refer to GFK-0075 which is the **Logicmaster 1 Family Programming Manual**.

How to Use This Chapter

This chapter is divided into 3 sections. Section 1 is a description of the requirements for programming. Included, are lists of all of the programming functions and a table of programming references assigned to each slot for 8 point modules. The purpose and types of references are explained, including “real world” use and internal use. A basic explanation of the scanning operation of the Series One and Series One Plus PCs is given. The scanning process is the basis for operation of all PCs and the user should have a good working knowledge of this operation.

The next 2 sections are devoted to programming. Section 2 describes and gives examples of the basic ladder diagram functions. Section 3 provides the user with a description of the data operation functions for use with a Series One Plus PC. The data operations greatly extend the number and complexity of applications possible with the Series One family.

Table 5.1 lists the basic ladder diagram functions for the Series One and Series One Plus. A more detailed explanation of each instruction can be found later in this chapter. Execution times in Table 5.1 are for Series One Model E, Series One Plus, and Plus 3.7K only. Execution times for Series One are listed in Table 5.2. Note that in the table, I/O = I/O points, IR = Internal Relay, SR = Shift Register.

Table 5-1. Series One Model E/One Plus Basic Ladder Diagram Instructions

Instruction	Definition	Memory Words	Execution Time (microseconds) For Series One Plus	
			Inactive	Active
STR	START. Start rung with a N.O. (Normally Open) contact.	1	6.6 50.9	6.6 I/O,IR,SR 50.9 Sequencer
STR TMR	START TIMER. Start rung with a N.O. contact referencing a timer.	1	10.3	10.3
STR CNT	START COUNTER. Start rung with N.O. contact referencing a counter.	1	10.3	10.3
STR NOT	START NOT. Start rung with a N.C. (Normally Closed) contact.	1	9.1 61.5	9.1 I/O,IR,SR 61.5 Sequencer
STR NOT TMR	START NOT TIMER. Start rung with a N.C. timer contact.	1	12.8	12.8
STR NOT CNT	START NOT COUNTER. Start rung with a N.C. counter contact.	1	12.8	12.8
AND	Add a N.O. contact in series with the previous contact.	1	5.3 6.2	5.3 I/O,IR,SR 59.1 Sequencer
AND TMR	AND TIMER. Add a N.O. timer contact in series with the previous contact.	1	5.3	5.3
AND CNT	AND COUNTER. Add a N.O. counter contact in series with the previous contact.	1	5.3	5.3
AND NOT	Add a N.C. contact in series with the previous contact.	1	8.4	8.4 I/O,IR,SR 60.3 Sequencer
AND NOT TMR	AND NOT TIMER. Add a N.C. timer contact in series with the previous contact.	1	8.4	8.4
AND NOT CNT	AND NOT COUNTER. Add a N.C. counter contact in series with the previous contact.	1	8.4	8.4
OR	Add a N.O. contact in parallel with the previous contact.	1	6.6 6.2	6.6 I/O,IR,SR 60.3 Sequencer
OR TMR	OR TIMER. Add a N.O. timer contact in parallel with the previous contact.	1	6.6	6.6
OR CNT	OR COUNTER. Add a N.O. counter contact in parallel with the previous contact.	1	6.6	6.6
OR NOT	Add a N.C. contact in parallel with the previous contact.	1	9.1 6.2	9.1 I/O,IR,SR 62.5 Sequencer
OR NOT TMR	OR NOT TIMER. Add a N.C. timer contact in parallel with the previous.	1	9.1	9.1

GEK-90842

Table 5-1. Series One Model E/One Plus Basic Ladder Diagram Instructions - Continued

Instruction	Definition	Memory Words	Execution Time (microseconds) For Series One Plus	
			Inactive	Active
OR NOT CNT	OR NOT COUNTER. Add a N.C. counter contact in parallel with the previous contact .	1	9.1	9.1
AND STR	AND STORE. Connects a logic group in series with the logic group preceding it.	1	3.8	
OR STR	OR STORE. Connects a logic group in parallel with the logic group preceding it.	1	3.8	
MCS	MASTER CONTROL START. Begin control of a block of logic with a master control relay.	1	5.0	
MCR	MASTER CONTROL RESET. Ends control of a block of logic with a master control relay.	1	3.0	
OUT	Defines a coil for a rung of logic, either an external output or an internal coil .	1	75	
SET	Defines a coil as being latched , the coil will remain on until turned off by RST.	1	17.5	
SET OUT	IF ON, the coil will remain ON even under control of the Output Disabler Coil (376).	1	10.0	
RST	RESET. Causes a coil that had been latched (SET) to be unlatched (Reset).	1	93	
SET OUT RST	SET OUT RESET. Defines a coil as being a one-shot (ON for one scan). (One Plus Only)	1	19.3	
TMR	TIMER . Programs a Timer operation. Times up from zero to a preset value. Elapsed time stored in an accumulate register. Preset time range is 0.1 seconds to 999.9 seconds.	2	27.1	700.0 I/O,IR,SR 458.8 Register
CNT	COUNTER. Programs an up counter. Counts up from zero to a preset value which can be 1 to 9999. The current count is stored in an accumulate register. Counters are retentive. Requires 2 rungs of logic; first rung enables the counter, the second rung resets the counter.	2	27.1	706.8 I/O,IR,SR 465.6 Register
SR	SHIFT REGISTER. Programs a shift register which can be 128 steps in length or a number of shift registers of varying lengths (128 bits total). Retentive. Requires 3 rings of logic; (1) data , (2) clock , (3) reset . can be programmed to shift forward or backward .	2	53.1	64.1 +16.6 For each bit shifted

Note: I/O = I/O Points, IR = Internal Relay, SR = Shift Register

Table 5.2 is a list of execution times for the Series One functions. Execution times are given in microseconds.

Table 5-2. Series One Execution Times

Instruction	I/O Point	Int Relay	Shift Register	Timer Counter	Sequencer	
					Inactive	Active
AND	33.9		32.7	37.5	36.0	84.0
AND NOT	36.5		35.1	39.9	40.8	88.8
OR	36.3		34.5	43.5	36	84.0
OR NOT	38.7		36.9	45.6	40.8	88.8
STR	40.8		39.6	48.0	89.4	89.4
STR NOT	42.0		43.2	50.4	97.2	97.2
SET	38.1		42.6			
RST	38.1		42.6			

Instruction	Inactive	Active
MCS	35.2	35.2
MCR	37.8	37.8
SET OUT	49.8	49.8
OUT	43.8	43.8
TMR	69.6	115.8
CNT	80.4	103.5
SR	55.2	66.9 + 12.9/N (1)
ANDSTR	34.2	34.2
OR STR	37.8	37.8

(1) N = number of bits shifted.

Table 5.3 is a list and description of the data operations that can be programmed, with a Series One Plus, in addition to the basic ladder diagram functions listed in Table 5.1. A more detailed explanation of each instruction is provided later in this chapter.

Table 5-3. Series One Plus Data Operation Instructions

Instruction	Definition	Memory Words	Execution Time (microseconds)	
			Inactive	Active
D•STR (F50)	DATA STORE. Loads accumulator with a binary or BCD value which can be a 4 digit constant or the contents of a specified 2 byte reference.	2	6.3	321.9 I/O,IR,SR 80.7 Register 14.3 Constant
D•STR1 (F51)	DATA STORE 1. Loads the lower byte (least significant) of the accumulator with the contents of a specified 1 byte reference.	2	6.3	140.9 I/O,IR,SR 63.8 Register
D•STR2 (F52)	DATA STORE 2. Loads the lower 4 bits of the accumulator with the upper 4 bits of a specified 1 byte reference.	2	6.3	172.2 I/O,IR,SR 95.0 Register
D•STR3 (F53)	DATA STORE 3. Loads the lower 4 bits of the accumulator with the lower 4 bits of a specified 1 byte reference.	2	6.3	173.8 I/O,IR,SR 96.6 Register

GEK-90842

Table 5-3. Series One Plus Data Operation Instructions - Continued

Instruction	Definition	Memory Words	Execution Time (microseconds)	
			Inactive	Active
D•STR5 (F55)	DATA STORE 5. Loads accumulator with a binary or BCD value from the 2 byte contents of a 16 circuit (high density) Input Module.	2	63	325.0 16 Point Input Module
D•OUT (F60)	DATA OUT. The contents of the accumulator (2 bytes) are written to a specified 2 byte reference (Output, Internal Relay, Shift Register Coils, Data Register or T/C Accumulate Register).	2	63	329.4 I/O,IR,SR 52.6 Register
D•OUT1 (F61)	DATA OUT 1. The contents of the lower byte of the accumulator are written to a specified 1 byte reference (Output, Internal Relay, Shift Register or Data Register).	2	63	160.1 I/O,IR,SR 39.1 Register
D•OUT2 (F62)	DATA OUT 2. The contents of the lower 4 bits of the accumulator are written to the upper 4 bits of a specified 1 byte reference (Output, Internal Relay, Shift Register or Data Register).	2	63	116.0 I/O,IR,SR 39.8 Register
D•OUT3 (F63)	DATA OUT 3. The contents of the lower 4 bits of the accumulator are written to the lower 4 bits of a specified 1 byte reference (Output, Internal Relay, Shift Register or Data Register).	2	63	108.1 I/O, IR, SR 55.0 Register
D•OUT5 (F65)	DATA OUT 5. The contents of the accumulator (2 bytes) are written to a specified 2 byte reference . Must be a 16 circuit (high density) Output module.	2	63	365.3 16 Point Output Module
= < (F70)	COMPARE. The contents of the accumulator are compared to the contents of a specified 2 byte reference (I, IR, SR, or DR) or a 4 digit BCD constant. Internal coils turn on to reflect the result of the comparison. Acc > DataOn 772 Ace = DataOn 777 Acc < Data011 774	2	63	354.0 I/O,IR,SR 112.8 Register 57.0 constant
+ (F71)	ADDITION (BCD). The contents of the accumulator are added to a 4 digit BCD constant or the contents of a specified 2 byte reference which must be a valid BCD number (Input, Internal Relay, Shift Register or Data Registers).	2	63	698.0 I/O,IR,SR 456.8 Register 262.0 Constant
- (F72)	SUBTRACTION (BCD). The contents of a specified 2 byte reference (Input, Internal Relays, Shift Register, or Data Registers) or a 4 digit BCD constant are subtracted from the contents of the accumulator.	2	63	557.0 I/O,IR,SR 3 15.8 Register 275.0 Constant
X (F73)	MULTIPLICATION (BCD). The contents of the accumulator are multiplied by the contents of a specified 2 byte reference (Input, Internal Relays, Shift Register, or Data Registers) or a 4 digit BCD constant . The lower 4 digits of the result remain in the accumulator, the upper 4 digits are stored in registers 576 and 577 which is the auxiliary accumulator.	2	63	497 to 2851 (I/O,IR,SR) 290 to 2644 (Register) 223 to 2576 (Constant)

Table 5-3. Series One Plus Data Operation Instructions - Continued

Instruction	Definition	Memory Words	Execution Time (microseconds)	
			Inactive	Active
÷ (F74)	DIVISION (BCD). The contents of the accumulator are divided by the contents of a specified 2 byte reference (I, IR , SR , or DR) or a 4 digit BCD constant. The first 4 digits of the answer are stored in the accumulator and the remainder is stored in the remainder is stored in the auxiliary accumulator (registers 576 and 577).	2	63	1218 to 2851 (I/O, IR, SR) 742 to 2645 (Register) 720 to 2577 (Constant)
D•AND (F75)	DATA AND (Logic product). The contents of the accumulator are logically AND-ed with each corresponding bit of a specified 2 byte reference (I, IR , SR , or DR) or a 4 digit BCD constant. The result is stored in the accumulator.	2	63	345.0 I/O,IR,SR 103.7 Register 55.6 Constant
D•OR (F76)	DATA OR (Logic Sum). The contents of the accumulator are logically OR-ed with each corresponding bit of a specified 2 byte reference (I, IR , SR , or DR) or a 4 digit BCD constant. The result is stored in the accumulator.	2	63	345.0 I/O,IR,SR 103.7 Register 55.6 Constant
SHIFT RIGHT (F80)	FUNCTION 80, SHIFT RIGHT N BITS. Contents of the accumulator are shifted to the right by the specified number of bits (1-15). Positions from which bits were shifted are filled with zeros .	2	63	216 + 13.4/N of bits shifted
SHIFT LEFT (F81)	FUNCTION 81, SHIFT LEFT N BITS. Contents of the accumulator are shifted to the left by the specified number of bits (1-15). Positions from which bits were shifted are filled with zeros .	2	63	220 + 13.4/N of bits shifted
DECODE (F82)	FUNCTION 82, DECODE. Decodes the lower 4 bits of the accumulator to a decimal number from 0 to 15 and places a 1 in the bit position in the accumulator which corresponds to that decimal number.	1	63	56.3
ENCODE (F83)	FUNCTION 83, ENCODE. Encodes a bit in the accumulator to a binary code representing the position number (0 - 15) and places the binary code in the lower four bits of the accumulator.	1	63	282.0
INV (F84)	INVERT. The 16 bits of the accumulator are inverted; i.e., 1 to 0 and 0 to 1. Example: 1101 1001 0110 0111 inverted to 0010 0110 1001 1000	1	63	30.3
BIN (F85)	BINARY. Converts a BCD value in the accumulator to a binary code.	1	63	412.2
BCD (F86)	BINARY CODED DECIMAL. Converts a binary code in the accumulator to a BCD value.	1	63	746.0
INTERNAL FAULT DIAGNOSIS (F20)	FUNCTION 20, EXTERNAL FAULT DIAGNOSIS. Allows external &vices to be monitored for unwanted conditions. If a fault occurs a preassigned 4 digit BCD number is displayed on the programmer. Multiple fault conditions can be monitored.	1	63	355.3 I/O,IR,SR 114.0 Register 722 Constant

Programming Fundamentals

This section provides a guide for entering and using each of the instructions. The instructions are presented in groups that correspond to their functionality. Table 5.4 lists the instructions grouped in this manner.

Table 5-4. Series One/One Plus Instruction Groups

BASIC INSTRUCTIONS	
	STR, STR TMR, STR CNT, STR NOT, STR NOT TMR, STR NOT CNT AND, AND TMR, AND CNT, AND NOT, AND NOT TMR, ANDNOTCNT OR, ORTMR, ORCNT, ORNOT, ORNOTTMR, ORNOTCNT ANDSTR, ORSTR MCS, MCR, OUT, SET, SET OUT, RST, SET OUT RST (ONE PLUS ONLY) TMR, CNT, SR
DATA OPERATIONS	
SERIES ONE PLUS ONLY	
MOVE	D•STR (F50), D•STR1 (F51), D•STR2 (F52) D•STR3 (F53), D•STR5 (F55) D•OUT (F60), D•OUT1 (F61), D•OUT2 (F62) D•OUT3 (F63), D•OUT5 (F65)
ARITHMETIC	>=< (F70), + (F71), - (F72), x (F73), ÷ (F74)
LOGICAL	D•AND (F75), D•OR (F76), INV (F84)
CONVERT	BIN (F85), BCD (F86), Shift Right (F80) Shift Left, (F81) Decode (F82), Encode (F83)
SPECIAL	External Fault Diagnosis (F20)

Significance of References

Whenever programs are entered into any PC, they must be accompanied by reference numbers. These references help to tell the CPU which function is specified. For example, which pushbutton controls the starting of which motor? Which timer are you referring to? Reference numbers are a vital part of programming, and in the Series One and Series One Plus are octal based; that is, they start at 0 and go up to 7, then jump to 10; at 77 the next value is 100. They look like traditional decimal numbers, except the digits 8 and 9 do not exist. Table 5.5 summarizes the various reference values and their significance.

Table 5-5. Summary of References

Octal Values	References	Decimal Quantity
000-157	I/O points (Series One)	112 total
000-157	I/O points (Series One Plus)	168 total
700-767	(Series One Plus only)	
770-777	Special Function Coils (Series One Plus only)	8
160-337	Internal Coils (Non-retentive)	112
340-373	Retentive Coils (Internal)	28 (1)
374-377	Special Functions	4 (1)
400-577	Shift Registers	128 (1)
600-677 (4)	Timers and/or Counters	64 (2) (1)(2)
400-577	Data Registers, 16-bit (Series One Plus only)	64 (3) , 124 Total if T/C references used as data registers

1. Retentive upon power failure.
2. Total maximum number of Timers and/or Counters.
3. Shift register and data register references are identical however, shift registers operate on bits, while data registers (located in a different area of memory) operate on bytes.
4. T/C references 600-673 can be used as data registers, if they are not referenced as timers or counters.

Significance of Input/Output References

I/O modules can be placed in any mix desired; however, once installed the exact reference used in programming is established by their physical placement. In other words, references are assigned to each location by the operating system in the CPU, and therefore **are fixed** for each location. As shown in the example in Figure 5.1, the modules are numbered **from** zero (adjacent to CPU) towards the **left** through all racks until the last module is reached at the left of the last rack. The exact reference is obtained by appending the circuit number (**0-7**), to the module number. Circuit numbers depend upon the terminal to which the field device is wired as discussed in Chapter 6. Table 5.6 summarizes available **I/O** references for 8 circuit modules for your convenience. This table can be reproduced locally to form a part of your system documentation. Notice that whether reference 005 is an input or an output depends upon which module is inserted into the **I/O** slot adjacent to the CPU. Refer to the Series One family compatibility guide in Appendix C for additional valid **I/O** references per physical location of modules.

GEK-90842

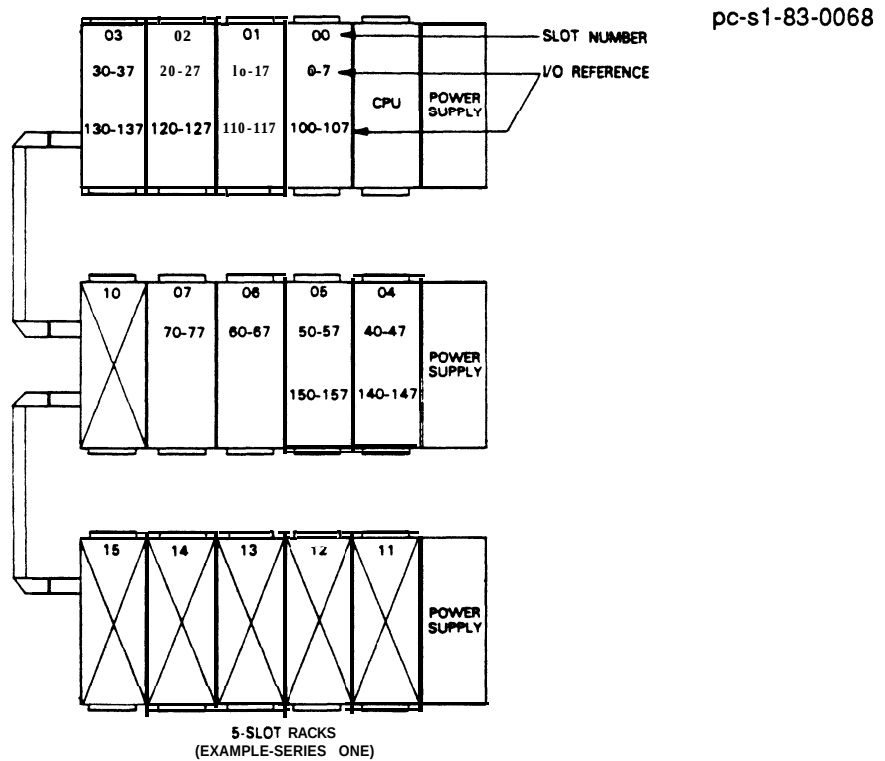
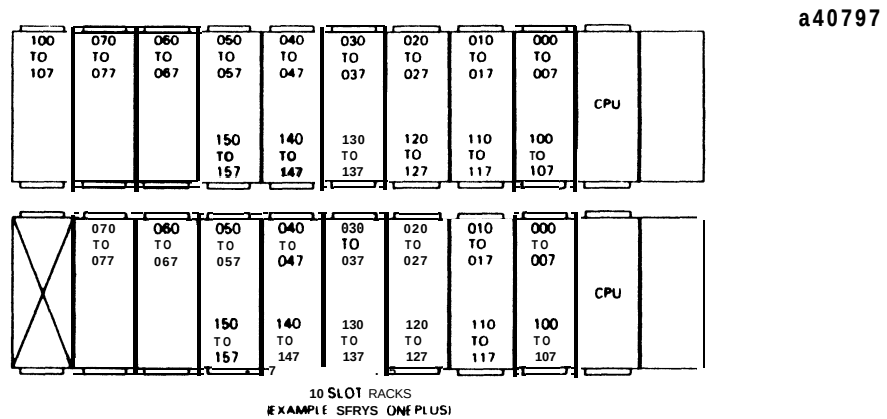


Figure 5-1. I/O References Per Physical Placement for 5 and 10-Slot Racks

NOTE

The addressing of slot 10 in a **10-slot** rack is determined by configuration of the two bridge connectors on the rack backplane.

Table 5-6. Summary of I/O References for 8 Circuit Modules

Module No.	Circuit No.	Reference	In/Out	Device Name	Wire No.
00	0	000			
	1	001			
	2	002			
	3	003			
	4	004			
	5	005			
	6	006			
	7	007			
01	0	010			
	1	011			
	2	012			
	3	013			
	4	014			
	5	015			
	6	016			
	7	017			
02	0	020			
	1	021			
	2	022			
	3	023			
	4	024			
	5	025			
	6	026			
	7	027			
03	0	030			
	1	031			
	2	032			
	3	033			
	4	034			
	5	035			
	6	036			
	7	037			
04	0	040			
	1	041			
	2	042			
	3	043			
	4	044			
	5	045			
	6	046			
	7	047			
05	0	050			
	1	051			
	2	052			
	3	053			
	4	054			
	5	055			
	6	056			
	7	057			
06	0	060			
	1	061			
	2	062			
	3	063			
	4	064			
	5	065			
	6	066			
	7	067			

GEK-90842

Table 5-6. Summary of I/O References for 8 Circuit Modules • Continued

Module No.	Circuit No.	Reference	In/Out	Device Name	Wire No.
07	0	070			
	1	071			
	2	072			
	3	073			
	4	074			
	5	075			
	6	076			
	7	077			
10	0	100			
	1	101			
	2	102			
	3	103			
	4	104			
	5	105			
	6	106			
	7	107			
11	0	110			
	1	111			
	2	112			
	3	113			
	4	114			
	5	115			
	6	116			
	7	117			
12	0	120			
	1	121			
	2	122			
	3	123			
	4	124			
	5	125			
	6	126			
	7	127			
13	0	130			
	1	131			
	2	132			
	3	133			
	4	134			
	5	135			
	6	136			
14	0	140			
	1	141			
	2	142			
	3	143			
	4	144			
	5	145			
	6	146			
	7	147			
15	0	150			
	1	151			
	2	152			
	3	153			
	4	154			
	5	155			
	6	156			
	7	157			

Table 5-6. Summary of I/O References for 8 Circuit Modules - Continued

Module No.	Circuit No.	Reference	In/Out	Device Name	Wire No.
70	0	700			
	1	701			
	2	702			
	3	703			
	4	704			
	5	705			
	6	706			
	7	707			
71	0	710			
	1	711			
	2	712			
	3	713			
	4	714			
	5	715			
	6	716			
	7	717			
72	0	720			
	1	721			
	2	722			
	3	723			
	4	724			
	5	725			
	6	726			
	7	727			
73	0	730			
	1	731			
	2	732			
	3	733			
	4	734			
	5	735			
	6	736			
	7	737			
74	0	740			
	1	741			
	2	742			
	3	743			
	4	744			
	5	745			
	6	746			
	7	747			
75	0	750			
	1	751			
	2	752			
	3	753			
	4	754			
	5	755			
	6	756			
	7	757			
76	0	760			
	1	761			
	2	762			
	3	763			
	4	764			
	5	765			
	6	766			
	7	767			

Internal Coils

The internal coils are control relays that can be used to control logic in the Series One or Series One Plus user program. However, their status cannot be provided directly to the I/O section. Retentive relays are specialized internal coils that have their ON or OFF status retained during time periods that the PC is not operating such as during loss of AC power. They use dual operations (set and reset) similar to hard-wired latching relays. A switch (**S1**) on the CPU module must be set to the ON position in order for the internal coils, 340 to 377, to be retentive upon loss of power. If the switch is not set properly, these coils will function as non-retentive coils.

Use of the Special Function Coils

Internal coils 374 through 377 are special purpose coils, in that they always perform specific internal functions and also provide useful internal system status. Coil 374 is a **first** scan reset or power-up indicator. Coil 375 provides a convenient way to program a 0.1 second clock pulse. **Coil 376 can be** used to inhibit all hardware outputs, that is, **outputs that are** connected to user devices (programmed with an OUT **XXX** instruction). Finally, coil 377 is an indicator of the operating status of the back-up battery for the CMOS memory devices. This coil could be used as a contact to **turn** on an output connected to a light, bell, etc. as a visual warning of a low battery.

Shift Register References

A group of references (**400** to 577) are used by the shift register. A total of 128 stages are possible and they can be used as one large shift register or a group of smaller shift registers (for example, 3 @ 30 stages and 2 @ 19 stages) as long as the total number of stages does not exceed 128. Additional details on all these functions will be provided later in this chapter.

Timer and Counter References

Another group of **fixed** references (**600** to 677) are used with timers and counters. Any mix of timers or counters totaling 64 can be used in a Series One or Series One Plus PC. Counters provide an additional function, in that they can also control a sequencer with up to 1000 steps. Counters are retentive upon power failure. References 600 to 673 can be used as **16-bit** data registers if they are not used as T/C references. T/C references 674-677 are reserved for Thumbwheel Interfaces or Timer/Counter Set Point use only.

Data Registers

Data registers are available **only with** a Series One Plus PC. This is a group of sixty four **16-bit** registers which can be used for storage of numerical data. Data can be written to or read from the data registers for use by the data operation instructions. Additionally data stored in the **registers** can be used as Timer/Counter presets. The references to be assigned to these registers are 400 to 577. Note that these references are identical to the shift register references. However, shift registers operate on individual bits (**400, 401, 402**, etc.), while the data registers operate on bytes and are located in a totally **different area** of memory. A byte in the Series One Plus PC is a group of 8 consecutive bits and each register is 2 bytes in length. By using available T/C references, the total number of data registers references can be up **to 124**.

Flexibility in Using References

Although references are assigned to specific functions as shown in Table 5.5, there is some flexibility in their use. I/O references not used by the hardware I/O section (for example, no module inserted into that slot or circuit not wired up) can be used as internal coils. However, if expansion is planned for the future, sufficient references to support that expansion should be reserved and not used in the current programming. Retentive coils can also be used as internal coils if their retentive function is not **required**. However, typically the unique value of retentive relays does not justify their sacrifice for a few internal coils. The shift register references can be used as a source for additional retentive or internal coils if they are not used for shift registers or sequencers.

Operating Principles

Before discussing the details of programming and the use of references, some details on the internal operation of the CPU can be useful. The majority of Series One and Series One Plus applications can be solved and the programs developed without consideration of the internal structure of the CPU. However, some applications can be more efficiently solved if knowledge of the CPU operation is applied while the program is being developed. The basic operation of virtually all PCs is referred to as a scanning function. There are many hundreds of decisions to be performed in any program, and the CPU cannot do all of them simultaneously. Similar to any electronic processor, it performs its operations one at a time. However, the speed of performance internally, makes the external results appear to have all operations accomplished at once.

Scanning

The term scan is a method of describing how the CPU performs its assigned tasks (see Figure 5.2). It begins at the **first** function, such as a relay contact, entered into the beginning of memory (address 0000). It proceeds sequentially through all memory addresses performing all functions entered by the user (for example, relay contacts, timers, latching relays, counters, sequencers, shift registers, etc.) until it reaches either the end of memory (address 1723) or the end of the program as entered by the user.

During the scan the logic sets or resets coils, according to the instructions entered into the logic program by the user. The status of these coils is immediately available to the next logic function. At the end of the logic scan, the CPU gets input data **from** the input modules and provides new data to output modules.

Next, the programmer, if connected, is serviced by making logic changes and/or updating its display. After servicing the programmer, the CPU performs a check of its internal hardware and resets the watchdog timer. The watchdog timer is a hardware timer set at 180 **msec** (milliseconds) to ensure that **memory or internal** circuit faults do not cause the CPU to enter **an** endless loop because of hardware failure. If a scan is not completed at least once every 180 **msec** (typical maximum scan is 60 msec), **the** hardware will shut the CPU down turning outputs OFF. Finally, any forced I/O **will** be entered. With successful completion of the internal checks, the CPU goes back to the start of the scan and continues its scanning sequence with the logic function entered at address **0000**. This repetitive scanning operation is performed from the time power is applied to the CPU until it is removed.

Total scan time includes user logic solution time plus overhead. The overhead time includes monitoring functions, I/O updates and housekeeping tasks. The typical scan times listed in the specifications in Chapter 1 assume that only basic instructions have been programmed (AND, OR, STR, NOT, etc.), they do not include overhead. The **overhead** time is about the same (4 to 5 **mSec**) for Series One and Series One Plus. This time must be added to the logic solution time to obtain the total real scan time.

GEK-90842

In order to achieve the maximum scan time, the monitor function should be disabled for both Series One and Series One Plus programs, since the monitor function adds about 2 **mSec** of overhead time when it is active. Also the overhead of a Series One Plus is increased slightly when communicating through the DCM, since both the Series One Plus CPU and DCM have been designed to provide a much higher throughput than the Series One **CPU/DCM** combination. Additionally, timer and counter instructions in the Series One Plus are slower than in the Series One because of the ability to use registers for preset values.

The scanning operation is very basic to PC operation and should not be overlooked. It provides a very useful verification of the CPU's reliability. For details on troubleshooting, refer to Chapter 7. The scanning operation also provides a fixed and definable sequence of logic decisions. Functions are solved in the order programmed. The results of one function (for example, coil, timer, shift register, etc.) are immediately available internally to the next logic element. In larger complex logic programs, internal races can be eliminated. The user does not need to be concerned with the time delay relays required to **pull in** or activate nor their variations (for example, a 4 pole vs 8 pole relay).

pc-s1-83-0004

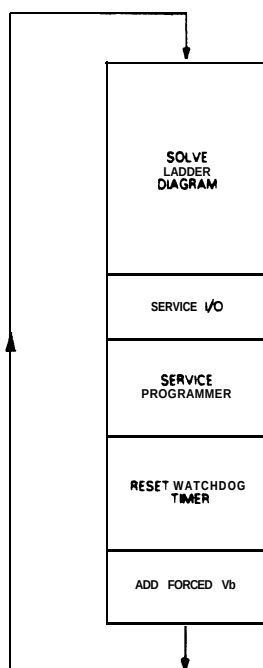


Figure 5-2. CPU Scanning Sequence

Programmer Functions

The basic programming language of the Series One and Series One Plus PCs is a simple relay ladder representation based upon standard Boolean functions (AND, OR, and NOT). Throughout the discussion of programming, examples will be used to illustrate the function described. The illustration of the Programmer (Figure 5.3) will be used as a reference for these examples and specific key sequences will

be provided so that the user can follow and demonstrate the example. A short-hand notation will also be provided that is recommended as a simple method to document your program. A complete discussion of the programmer is provided in Chapter 4. A brief overview of important keys is presented below to aid the user's understanding of the examples.

The keyboard has both upper and lower case functions similar to a typewriter. The numerical keys for most programming steps are accessed by **first** selecting the SHF (Shift) key to the upper right. Once depressed, the SHF LED is lit (display center right) and then the appropriate numerical digit(s) can be selected. The shift function is latched and will be released only by selection of the ENT (Enter) or CLR (Clear) key. The SHF key does not have to be held down. The display window in the upper left reflects either address or data information (not both) as the operation progresses. Typically, as keys are selected, their **respective LEDs** will light to display the function selected. Other important keys are as follows:

KEY	FUNCTION
AND	AND referenced status with previously entered logic (Series)
OR	OR referenced status with previously entered logic (Parallel)
STR	START new rung of a ladder diagram
NOT	NOT or invert (i.e. make normally closed) contact referenced
OUT	OUT put logic status to a coil, either output or internal
TMR	TIMER is selected as desired function
CNT	COUNTER is selected as desired function
SET	SET latching relay or shift register status
MCS	Master Control Start • begin control of master control relay
MCR	Master Control Reset • end control of master control relay
SR	Shift Register • establish range of shift register
RST	RESET latching relay or shift register status
F	FUNCTION • specifies a data operation (Series One Plus)
R	REGISTER • specifies a data register or group entry (Series One Plus)

a41 939

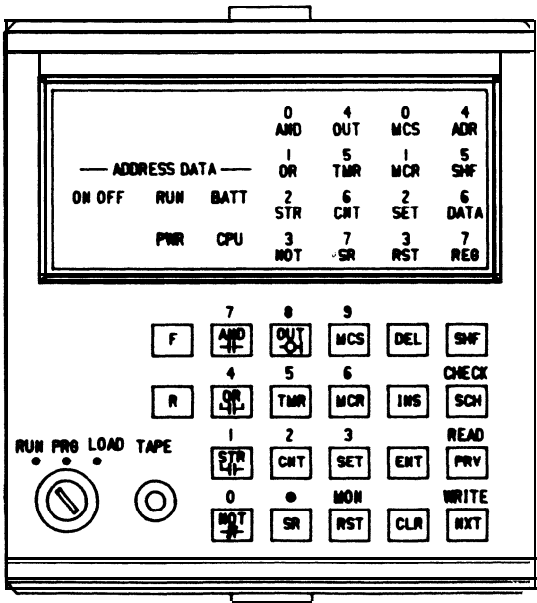


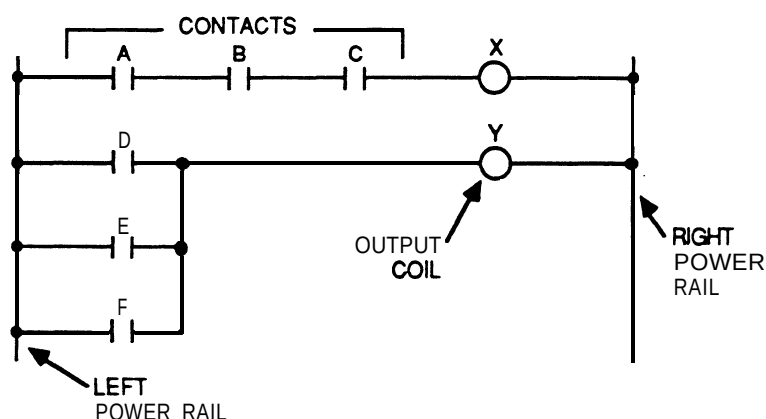
Figure 5-3. Programmer for Series One and Series One Plus

GEK-90842

Basic Ladder Diagram Format

Figure 5.4 illustrates a typical ladder diagram. Between two vertical power rails to the extreme left and right, contacts are placed in horizontal strings or lines. Adjacent lines can be connected between contacts by a vertical line to allow logic to be solved in parallel. The horizontal strings (series) of contacts **are** equivalent to **ANDs**. For example line 1 can be described as: A and B and C must occur before coil X is energized; any one reference can prevent the coil from energizing. Similarly vertical contacts are in parallel and programmed as **ORs**. Thus, line 2 is described as: D or E or F will energize coil Y. Any one reference can by itself energize coil Y.

In the Series One and Series One Plus PCs, there is no internal limit on how many contacts can be **placed in** series, nor how many in parallel. However, as a practical limit for simple programming **and** system documentation, it is recommended that a horizontal string be limited to nine contacts and one coil; and a vertical array to seven parallel lines.



a42342

Figure 5-4. Typical Ladder Diagram

Concept of Power Flow

One key feature of PCs is power flow. This is a conceptual flow of power used to visualize the operation of coils, timers, counters etc. Referring again to Figure 5.4, the left hand power rail can be envisioned as “hot” (connected to 115 V ac or +24 V dc) and the right as its associated “neutral” (connected to 115 V ac or -24 V dc). **The coils (e.g. X and Y) will be** energized if there is a path for power flow from the left leg to the coil placing the full potential across the coil. If there is no power (electron) flow, the coil will be de-energized (OFF).

Power **always flows from** the left towards the right and will pass through normally open contacts if their references **are** energized (ON) or **normally** closed with de-energized (OFF) references. Power flow is also allowed vertically either up or down between adjacent lines where **parallel connections are programmed**. However, power can NOT flow from right to left at any time through contacts or horizontal shunts. **This** feature **simplifies programming** and prevents undesired sneak **paths**. However, if hard wired **relays are replaced** by a Series One or Series One Plus PC, it is possible some adjustments may be necessary to the logic to either simplify the **programming** or **to** add sneak paths that may be a **basis for the relays functioning**.

Unlimited References

Another difference between PC **programming** and hardwired relays is that any reference such as an input or coil can be used on relay contacts as often as necessary. Since references are merely a unique series of bits in a word of memory, they can be programmed wherever and whenever necessary. Now with PCs, there is no need to count relay contacts and try to limit individual references to four or less contacts; no need to plan to use form C relay configurations to squeeze two contacts **from** one pole. Any reference can be envisioned as a relay controlling many poles providing both normally open and normally closed contacts **that** operate without any time delay from pole 1 (closest to the coil) to pole 1000 (furthest away from coil).

GEK-90842

SECTION 2

Basic Instructions

Basic Instructions

This group can be considered as the basic building blocks for a relay ladder diagram. The instructions reference discrete bits that are to be part of an operation, that is, a conditional contact or the end of a rung, which could be an output or an internal relay.

How to Begin Programming

The easiest way to begin feeling comfortable with **programming** a Series One or Series One Plus PC, is to enter one rung of logic with contacts in series that control a single coil. Physically connect input devices (pushbutton switches, limit switches, etc) and an output device (such as a lamp) to the corresponding terminals on an input and output module in the PC, turn the inputs on and off, and observe the result on the output device.

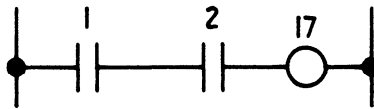
Entering a Rung With Series Contacts

In the following program sequence, a rung of ladder logic is shown, then the keystroke sequence required for entering the logic. Notice that when entering a numerical sequence, the SHF key must be depressed before the number or number sequence. In the example, each keystroke is separated by a comma. Numbers with more than 1 digit are grouped together for ease of interpretation.

Turn the mode keyswitch to the PROG (**programming**) position in order to enter the logic.

Enter **two** normally open contacts (1 and 2) in series controlling the state of an output coil (17). Both contacts must be closed to tum on the output.

a40347



When input devices **wired** to inputs 1 and 2 are closed, output 17 will tum on. The operating state of the inputs and outputs, either ON or OFF, can be monitored by observing the **LEDs** on the Programmer's logic display that **correspond** to the input or output.

Entering the Clear all Memory Sequence

After observing the operation of the series rung, the rung can be deleted by entering the clear all memory sequence, or the rung can remain in memory, with the next rung starting **at** the next memory address. The next memory address, assuming that **the** above rung was entered at address 0000, is **0003**.

If you wish to clear tie contents of memory and start the next rung at address 0000, enter the following key sequence.

CLR, SHF, 348, DEL, NXT

When the clear all memory sequence has been successfully entered, the rung of logic that had been entered will be cleared from memory. (Remember, whenever the clear all memory sequence is entered, **the** entire contents of memory will be cleared).

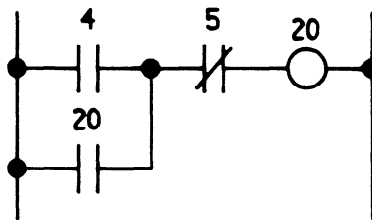
NOTE

Be sure that you want to clear the complete contents of memory when using this sequence, there is no recovery.

Entering a Rung with Parallel Contacts

Continue the simple program entry by entering a normally open contact (4) in parallel with a second **normally** open contact (20) that references the output coil (20). In series with these, add a normally closed contact (5), then the output coil.

a40347

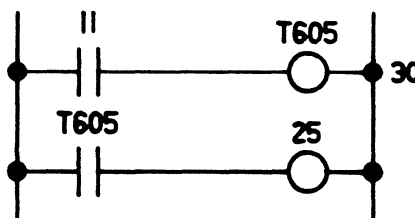


If an input device, such as a limit switch is wired to input 4, and the limit switch is closed, the output would turn on. The normally open contact referencing the output would close and become a seal contact, which would maintain the output in the on state, even if the limit switch wired to input 4 were then opened. The output could be turned off, in this example, by momentarily closing a switch connected to the closed contact, reference 5. Since this is a normally closed contact, it would open when the switch is depressed, and the output would turn off. This logic is commonly used as a motor starter.

Entering a Simple Timer Rung

Next, enter two rungs of logic, the first ending with a timer, having a preset value of 30 **seconds**. The second **rung** has an output controlled by the state of the timer coil. When input 11 is closed, the timer begins **timing** up from 0 towards the preset value of 30 seconds. When the accumulated value of the preset reaches 30, the timer coil will turn on, contact **T605** will close, and output 25 will turn on.

a40347



GEK-90842

Basic Relay Logic (Motor Starter)

With this background on the basics of programming a Series One and Series One Plus PC, Figure 5.5 provides a simple example to illustrate relay ladder programming. The references used were selected for illustrative purposes and can be adjusted as necessary to meet your application needs. This is a simple motor starter and seal circuit. An input module is assumed to be installed next to the CPU (005 = start pushbutton input, and 006 = stop pushbutton). Adjacent to the input module is assumed to be an output module (010 = output to motor starter). The program does not care whether these input and output modules are 115 V ac or 24 V dc. The programmer keys to be depressed to enter this logic are also shown in the figure. The start key begins a new rung of logic; this example and most that follow assume an empty memory so the example begins at address 0000.

pc-s1-83-0006

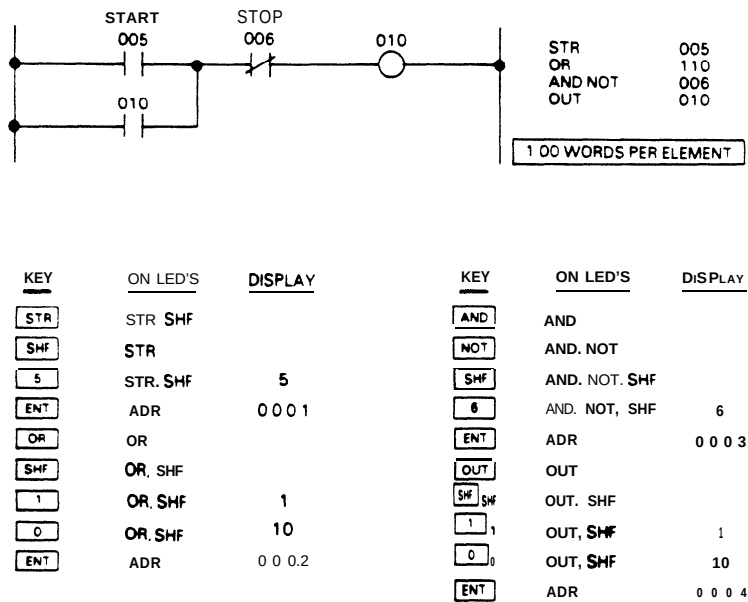


Figure 5-5. Sample Relay Logic (Motor Starter)

Motor Starter Logic Description

The first reference is entered (STR, SHF, 5, ENT) and a normally open contact is established since the NOT key was not depressed. Referring to Figure 5.5, the next function moving from the left (contact 005) to the right along the top line of the logic rung, is a parallel contact referenced as 010. Parallel logic is represented by the OR function, which is selected followed by the desired reference (10) and the Enter key to load the memory. Again moving to the right of the top line, the next contact is in series with the logic completed so far. Thus the ANTI key is selected (for series logic). Since this contact is normally closed, the NOT key is also selected followed by the reference 6 and the Enter key.

The final element of this logic is the coil or output of the rung. Selecting the Output key does not ensure a “real world” output will be generated; it also must refer to an I/O reference and have an output module inserted in the proper slot. To establish the rung output coil, the output key is selected followed by the Shift key, the reference (10) and the Enter key. Outputs can be paralleled by entering additional OUT functions immediately after the initial one.

Normally Closed Input

After completing the logic of Figure 5.5, several features of programming should be noticed. First, all elements of the program are entered directly into CMOS memory as the Enter (**ENT**) key is depressed. Thus, if power should be interrupted while a program is being entered or altered, it will be totally saved except possibly for the one element being worked on. Second, there is feedback on the programmer **for all** entries. Something happens (LED ON, display changes, error codes appear, etc) for each key selected. Third, the logic is set up for inputs that are **wired** normally open. If an input is wired normally closed, an adjustment must be made to the symbol (NO or NC) used for that input. Referring to Figure 5.6, inputs sense voltage at their field terminals. They can not detect if a voltage is from a normally open limit switch that is depressed or from a **normally** closed pushbutton that is not depressed. Cover up the two field devices wired to inputs 002 or 003. If you were the input module, and both applied voltage to the field terminals, could you tell which was wired normally closed and which normally open? Because of this, the programmer or system designer must compensate.

pc-s1-83-0007

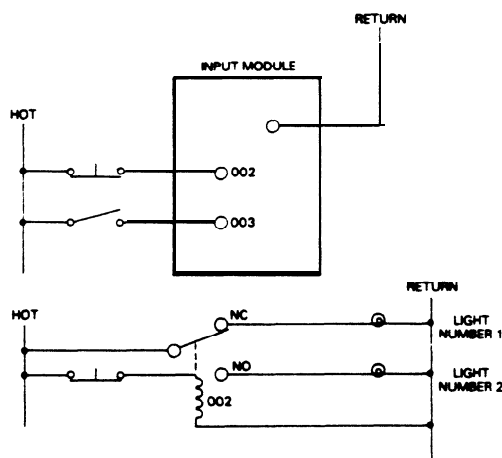


Figure 5-6. Example of Normally Closed Inputs

In a previous discussion, it was stated that inputs are like relays with many poles and you are selecting which type of pole is to be used when you program. In the lower half of Figure 5.6, coil 002 is shown in the de-energized position; however it is wired to a normally closed pushbutton. When you want power to flow if the pushbutton is **NOT** depressed, which contact type do you choose, NO or NC? The answer is normally open since coil 002 is always energized except when the pushbutton is **depressed**. **Therefore, in programming** with the Series One or Series One Plus PCs, a **normally** open contact is appropriate. If the stop button of Figure 5.6 **were wired normally closed, its internal logic** should be normally open to compensate. However, this concept does not normally cause problems for the programmer. **Since PCs are reprogrammable**, if an error is made and discovered when the system is exercised, it is easy to correct.

GEK-90842

Push-Down Stack

Two very useful functions are programmed using a **2-key** sequence, the AND STR and the OR STR. The AND STR provides a common series connection between a block of logic and the OR STR provides a common parallel connection for a block of logic. These instructions provide the connecting links not allowed with any of the previously mentioned instructions. The logic used with this type of operation is referred to as a push-down stack. The push-down stack can accommodate up to eight levels (groups) of logic. A push-down stack can be thought of as a temporary storage **area** to allow the combining of elements in series (AND) or parallel (OR) connections. **Figure 5.7** illustrates the use of these functions.

a40545

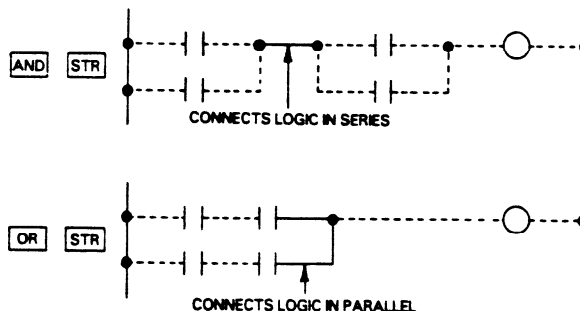


Figure 5-7. AND STR and OR STR Connections

The following pages contain a more detailed description of a pushdown stack and programming with the AND **STR/or STR** instructions.

Detailed Example of AND STR, OR STR

The push-down stack can be used for several groups of contacts, either in series or in parallel, and can be reused many times in the user's logic. As mentioned previously, the stack has eight levels or storage locations where an ON/OFF (1/0) condition is stored.

PUSH-DOWN STACK	
ON (1)	Location 1
ON (1)	Location 2
OFF (0)	Location 3
ON (1)	Location 4
OFF (0)	Location 5
OFF (0)	Location 6
ON (1)	Location 7
OFF (0)	Location 8

Figure 5-8. Push-Down Stack Storage Locations

Description of Operation

There are four basic rules that govern the operation of the push-down stack and they are explained in the following examples.

Rule Number 1: A STR X instruction pushes the contents of each position in the stack down one location. The contents of location number 1 are pushed down to location number 2, the contents of location number 2 are pushed down to location number 3, and so forth. The logical status of reference X is then moved into location number 1 of the stack.

Rule Number 2: The AND Y/OR Y instruction logically ANDs/ORs the status of reference Y (ON/OFF) with the ON/OFF status of location number 1 in the stack and writes the result back into location 1.

AND (SERIES)			OR (PARALLEL)		
Y	BEFORE LOCATION NUMBER 1	AFTER LOCATION NUMBER 1	Y	BEFORE LOCATION NUMBER 1	AFTER LOCATION NUMBER 1
OFF	OFF	OFF	OFF	OFF	OFF
ON	OFF	OFF	ON	OFF	ON
OFF	ON	OFF	OFF	ON	ON
ON	ON	ON	ON	ON	ON

Figure 5-9. Push-Down Stack Logical Operations

GEK-90842

Rule Number 3: The **AND STR** or **OR STR** instructions logically AND or OR location number 1 in the stack with location number 2 and writes the result into location number 1. In other words, these instructions put one group of contacts in series or parallel with another group of contacts. This instruction also moves the contents of positions **3** through **8** up one location.

Rule Number 4: An **OUT Y** instruction sets Y to reflect the status of location number 1 in the stack. If the status of location number 1 is **ON (1)**, output Y will be turned off. If the status of location number 1 is **OFF (0)**, output Y will be turned off.

pc-s1-83-0055

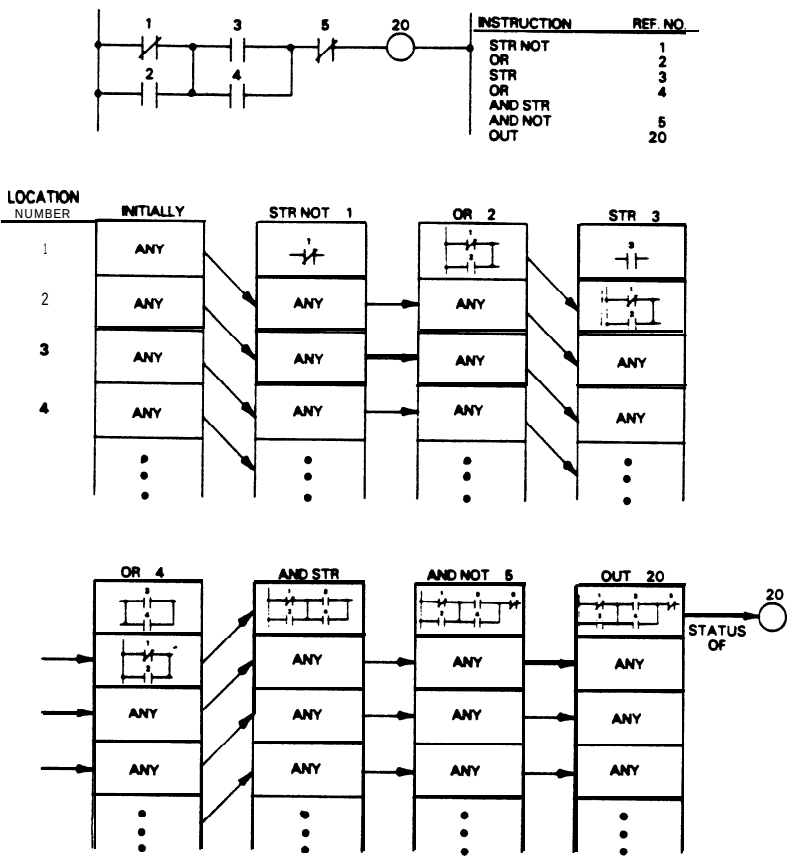


Figure 540. AND STR/OR STR Example Number 1

pc-s1-83-0056

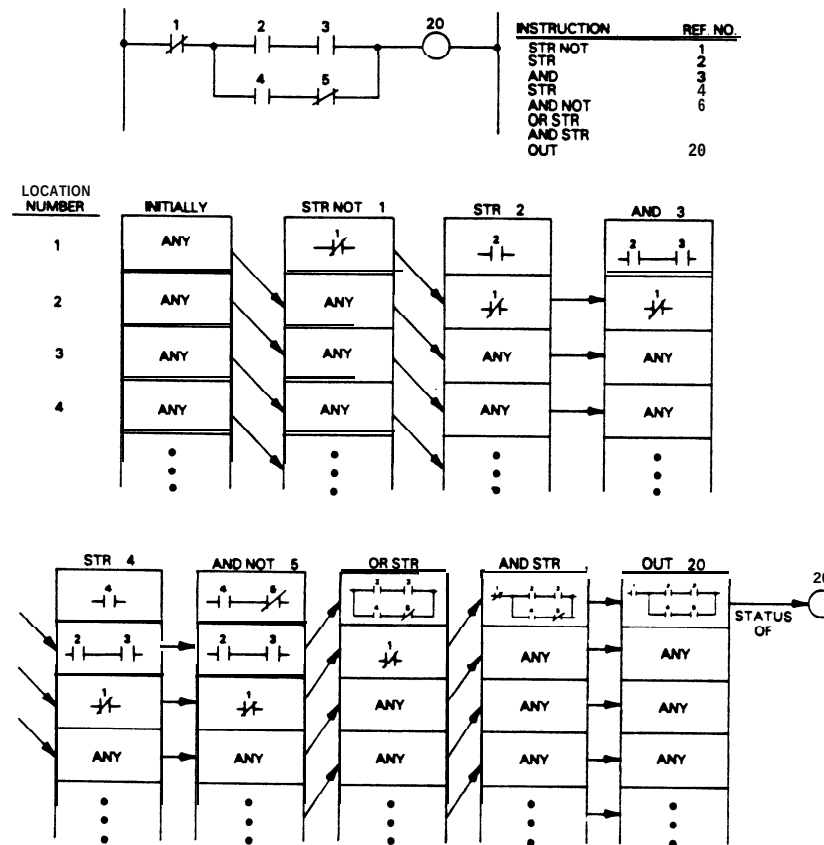


Figure 5-11. AND STR/OR STR Example Number 2

Use of Retentive Coils as Latches

All coils referenced so far have not been retentive upon power failure. Thus if a non-retentive coil is ON when power is removed **from** the CPU, it will be OFF upon reapplication of power (if Switch 1 on the CPU module is **ON**). **There are 28** special coil references (**340-373**) that **are** retained upon power failure. Thus if they were ON prior to loss of power, they will be ON when power is restored; they will be OFF while power is OFF since no scanning is being performed. In addition to these 28 references, latches can be built using shift register references (**400-577**) that are not being used to perform shift register functions. Either retentive coils (**340-373**) or shift registers (**400-577**) can be used to control contacts, both normally open and normally closed wherever required. **This** special retentive function is derived **from** their use as coils to complete or store the results of relay logic rungs.

Programming a Latched Relay

Latches require two separate functions or inputs similar to hard-wired relays, a **SET (turn ON)** and a **RESET (turn OFF)**. **These functions are commonly** referred to as latch and unlatch. If both are active, the last one in the scan will be controlling the state of the coil. To turn on a latch, the function SET

GEK-90842

followed by the retentive coil reference (for example, 340-373 or 400-577) is used instead of OUT, plus the coil number at the end of a logic rung. Once power flows to that coil, it will be energized (turned ON) and remain ON even if power flow to it should be interrupted.

To turn a latch off (de-energize) separate relay logic should be built ending in a RST (Reset) function and the [same] coil number. In many applications, it is useful to allow the system to clear (turn OFF) coils when power fails so that an orderly restart is possible when power is restored, whether that is 2 seconds or several hours, or several days later. Other applications require the control system to remember the status of key items such as part position, operations completed, elevators up or down, etc. The Series One and Series One Plus are provided with both standard and retentive coils to allow the user to tailor the program to fit their requirements.

Figure 5.12 illustrates the programming of retentive (latched) relays. Coil 340 is used only as an example. The logic to either set or reset a retentive coil can be of any convenient size similar to standard coil logic. When power flows through the top rung to the SET coil 340 function, it will be energized and remain ON even if power flow is removed. Anytime the power flows through the lower rung (regardless of path), coil 340 will be turned OFF. Since the set coil 340 is programmed before the reset, it is placed in earlier memory locations and the reset function will have overriding control if they are both energized.

pc-s1-83-0011

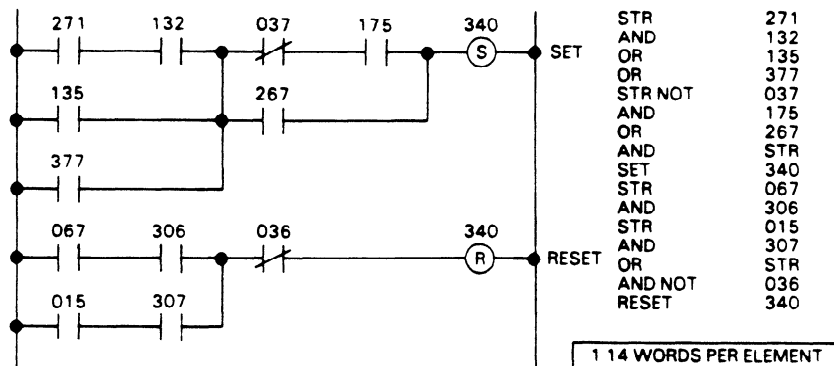


Figure 5-12. Example of Latching Logic

Master Control Relay Functional Description

To control large quantities of coils on a supervisory basis, a Master Control relay function is provided. This function allows an efficient method of programming to turn off large quantities of coils in case some major permissive is not satisfied. One approach would be to incorporate a permissive contact in each rung controlling one of the coils. This may require many contacts and inefficient use of memory. Another method is to bracket those rungs over which master control is desired; this is similar to quotation marks around a direct statement (one at each end to define exactly where the statement starts and where it ends).

The permissive logic is first built starting to the left of the ladder diagram adjacent to the "hot" power rail. Then an MCS (Master Control Start) function is entered, followed by the logic it is to control, and ending with an MCR (Master Control Reset) function. The MCS and MCR are like quotations around the logic to be controlled. When power flows to the MCS, the logic enclosed will operate normally. If

there is no power flow to MCS, the coils will be forced to the OFF state regardless of the internal logic conditions, timers will be reset to zero, and counters will stop (frozen) but not reset. The amount of logic controlled by the Master Control relay function is limited only by the memory provided.

pc-s1-83-0012

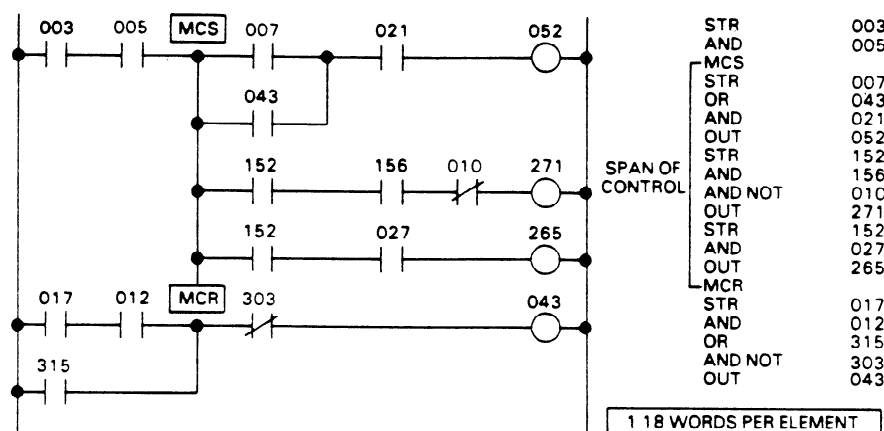


Figure 5-13. Example of Master Control Relay Logic

Programming a Master Control Relay Function

Figure 5.13 illustrates the use of one pair of MCS/MCR functions. In many cases it is convenient to view the MCS function as defining a sub-left power rail whose connection to the main power rail is dependent upon some relay logic. The conditional logic (I003 - I125 in this example) is built first using the normal relay logic. The MCS function is entered without any reference to create the beginning point of the control. Logic is built using normal functions within the control area. Note the duplication of contact 152, which cost only one memory word, there is no need to count or limit the quantity of relay contacts used in PCs. To end the master control, the MCR function is entered returning control to the next power rail to the left and conventional logic can now be built. Unless both references 003 and 005 are energized (ON) in this example, coils 052, 271, and 265 will be OFF. When these contacts are passing power, coils 052, 271, and 265 will respond to their normal logic.

Multiple Master Control Relay Functions

Multiple master control functions are possible in any logic program. They can be embedded within the scope of the first MCS and the last MCR as necessary, as illustrated in Figure 5.14. The first group of coils under the Master Control of references 010 [or] 011 is 204-213 which includes two smaller groups (207-210 and 212-213). Group 207 and 210 are under the Master Control of 010 or 011 and 015, while group 212 and 213 are controlled by 010 or 011 and 022. Note the requirement for two successive MCRs to end both the group 212-213 and the larger group 204-213. To operate correctly there must be an equal number of MCS and MCR functions in your program.

GEK-90842

pc-s1-83-0013

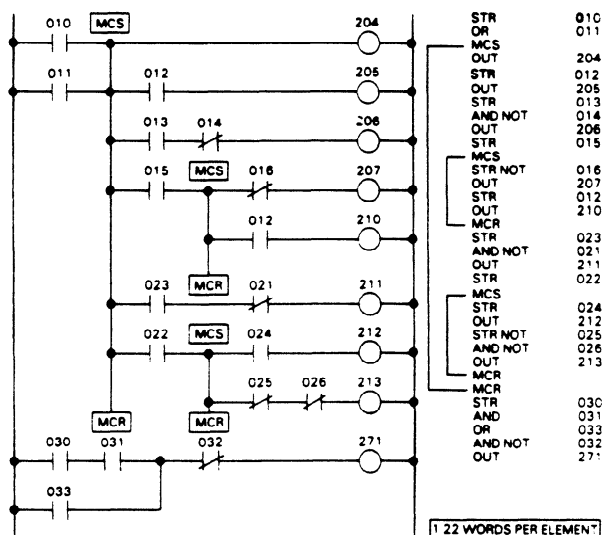


Figure 5-14. Multiple Master Control Relay Logic

Disabling of Outputs

In addition to Master Control functions, all outputs can be affected by an internal special function coil referenced as 376. When this coil is energized by user logic, all outputs in the hardware I/O structure will be turned OFF. Internal coils and statuses will still operate, but only internally. The only exception is to coils that are programmed as SET OUT functions rather than the OUT function. These will still be able to provide their normal output status (ON or OFF) to the I/O section.

Figure 5.15 illustrates the use of coil 376. In this example, when either reference 003 or 056 (inputs, outputs, or internal depending upon hardware I/O configuration) is energized, coil 376 will also be energized. This will cause all coils programmed with the OUT function, such as 010, to be turned OFF at their output module, although they will function normally internally. However, coils driven by the SET OUT function, such as 011, will not be affected by the status of coil 376. Table 5.7 summarizes the effect of coil 376 on the OUT and SET OUT functions.

NOTE

It is not recommended that the Output Disabler coil (376) be used with retentive coils (latches).

Table 5-7. Effect of Coil 376 (Output Disabler)

Internal Status	Coil 376 OFF		Coil 376 ON	
	Out	Set Out	Out	Set Out
OFF	OFF	OFF	OFF	OFF
ON	ON	ON	OFF	ON

pc-s1-83-0014

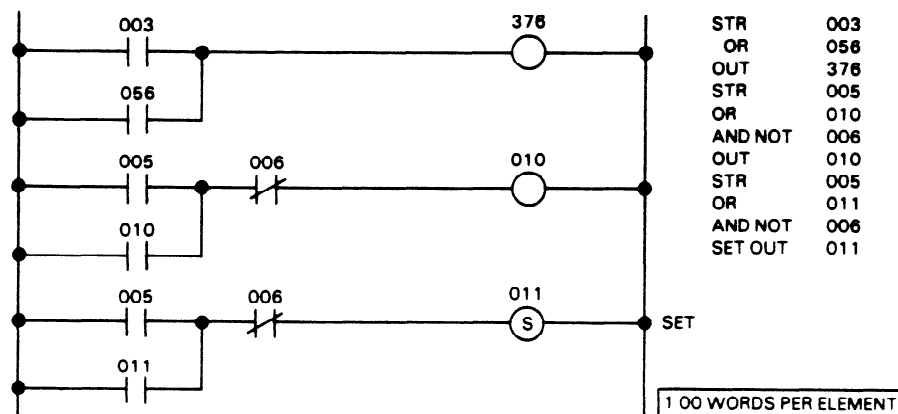


Figure 5-15. Sample Coil Disabler Logic

Timer and Counter Functional Description

In addition to relay logic, there are a wide variety of functions that are standard with the Series One and Series One Plus. Timing and Counting are two of the most commonly used non-relay functions. These functions are special coils using references 600-677. Up to 64 total timers and/or counters can be built within the Series One or Series One Plus. They end a rung of logic similar to using an OUT function for relay logic. Unlimited contacts, both normally closed (timed OFF delay) and normally open (timed ON delay) can be referenced to any timer/counter coil. A preset value is used to indicate how many seconds, tenths of seconds, or counts, the function should delay before energizing its coil. When the coil is energized, time or counts continue to be recorded. All presets can be up to four decimal digits (0001 to 9999), which allow timers to be set from 0.1 to 999.9 seconds, and counters from 1 to 9999 counts. Timers can be set from 0.01 to 99.99 seconds with the Series One Plus 3.7K PC only. Timers are set to a .01 second duration by setting output reference 770 (SET 770). Caution must be used when doing this, since reference 770 being reset in the same program would cause the timers to revert back to .1 second duration.

Special Timer/Counter References

The four references 674-677 obtain their presets from external thumbwheels via the thumbwheel interface. Additionally, a Timer/Counter Setpoint unit can be used with the Series One Plus to enter the preset values. When a preset value is selected with the Timer/Counter Setpoint unit, the values are simultaneously entered into 8-bit registers, (two registers for each 4-digit BCD value). The registers can then be used to specify the preset value for any Timer or Counter. It should be noted that only 1 Thumbwheel Interface unit or 1 Timer/Counter Setpoint unit can be used with a series One Plus PC, not both at the same time. These four references cannot use internal presets; however, they will count up to 9999 and can be used to drive sequencers without the thumbwheel interface.

Programming Timers

Timers (Figure 5.16) can be programmed in either seconds or tenths of seconds (or hundredths of seconds with a Series One Plus 3.7K PC) and require only one line of logic to control their operation. When power flows to the timer coil, it will begin recording time at the rate of one count every 0.1

GEK-90842

seconds or .01 seconds as applicable. The timer starts at 0000.0 and records time, towards the preset value. When the preset value is reached, an event will happen according to what has been programmed. Also, when the timer reaches the preset value, the timer will continue timing up, until reset by an interruption of power flow through its conditional contacts, or until it reaches its maximum value of 999.9 or 99.99, as applicable at which time it will stop. If power flow to a timer is interrupted or if the CPU should stop operating, the timer will be reset to zero. The basic clock driving all timers derives its accuracy from a crystal within the CPU, it does not depend upon the AC power line frequency.

pc-s1-83-0015

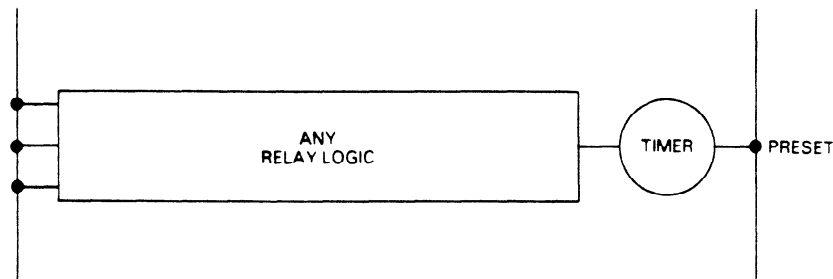


Figure 5-16. Sample of Timer Logic

A timer can be programmed to be self resetting, by programming a closed contact referencing the timer coil. If programmed in this manner, when the preset value is reached, the timer coil will turn on, the closed contact referencing the coil will open and the timer will reset to 000.0 and start timing up again.

Figure 5.17 illustrates the programming of various types of timers. The relay logic is built in the normal manner up to the coil selection. The timer (TMR) function is selected along with a timer/counter reference (600-677). Each timer or counter must have its own unique coil to operate properly. After the timer is identified, the preset is entered.

In this example, coil 41 will be OFF until 15 seconds after timer 601 is energized. It will remain ON until either the power flow to 601 is interrupted or the CPU goes through a power up operation. Coil 42 is inverted from 41 in that it stays ON until 15 seconds after power flows continuously to the timer 601, when it goes OFF. Timer 602 delays 5.5 seconds after reference 27 is energized before it turns its coil ON, also energizing coil 43.

Specifying Timer Preset Values

The Timer preset value can be either a fixed value entered after the T/C reference, the BCD value entered with the Timer/Counter Setpoint unit for the special Timer/Counter references (674-677), or the value in the registers that are assigned to the special Timer/Counter references. These registers are 564 and 565 for T/C reference 674; 566 and 567 for T/C reference 675; 570 and 571 for T/C reference 676; and 572 and 573 for T/C reference 677.

The preset value information is also valid for Counters.

a40016

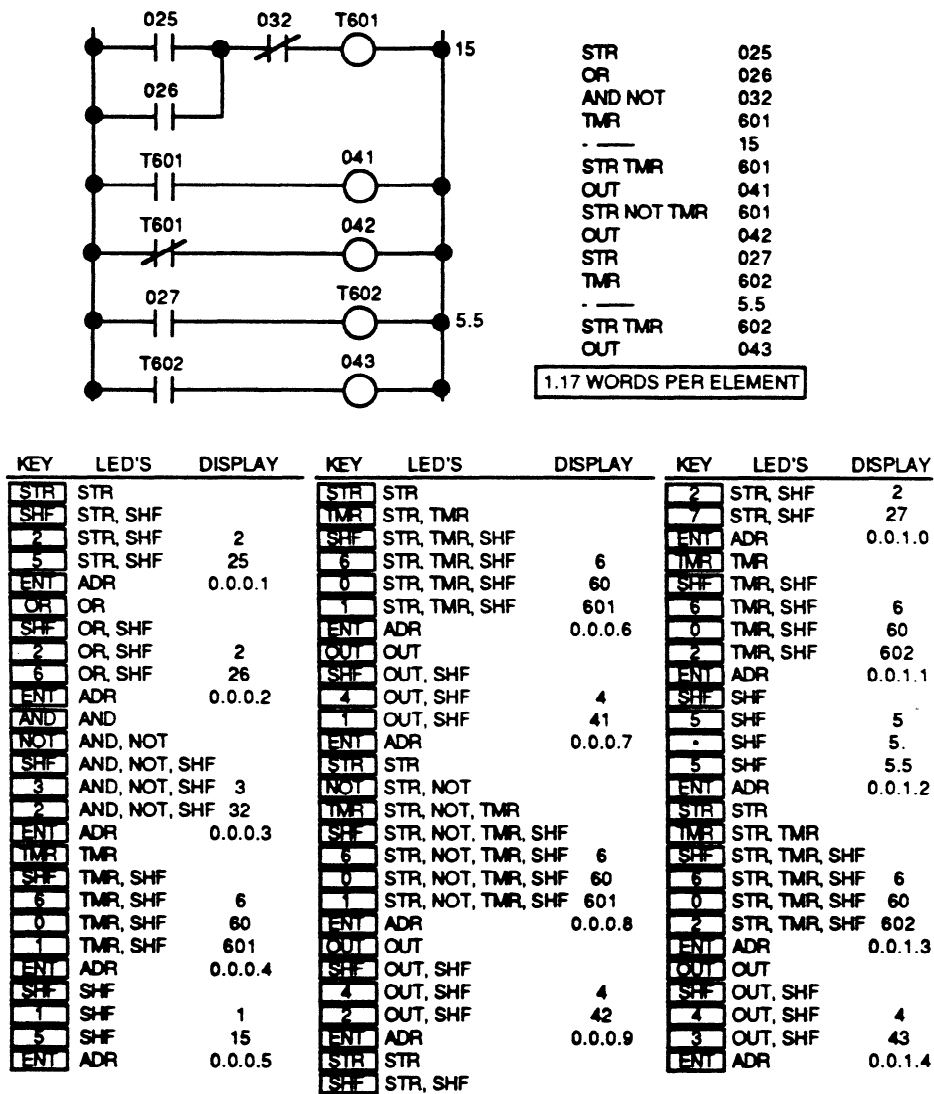


Figure 5-17. Example of Timer Logic

Programming Counters

Counters (Figure 5.18) operate similar to timers except that they require two rungs of relay logic to control their operation. The upper rung controls when the counter is incremented. When this rung goes from no power-flow to power-flow (OFF to ON), the counter is incremented by one. To cause another count to be recorded, power-flow must be interrupted and another OFF to ON transition must occur. All counters count up, starting at zero, towards a preset value. The ability to detect transitions and record counts is built into the counter function and requires no further programming by the user. The lower rung of logic (second STR function) controls the reset of the counter function. Whenever this rung supplies power flow to the counter, the counter will be reset to zero. If both rungs supply power flow, no

GEK-90842

counts are recorded and the counter is forced to zero. All counters are retentive upon loss of CPU power.

pc-s1-83-0017

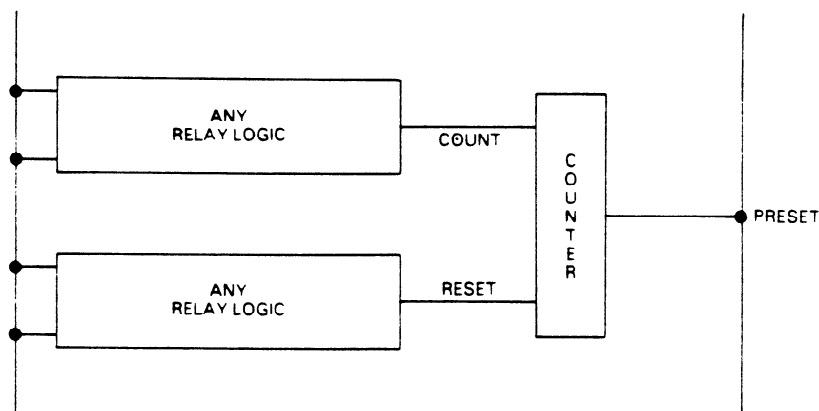
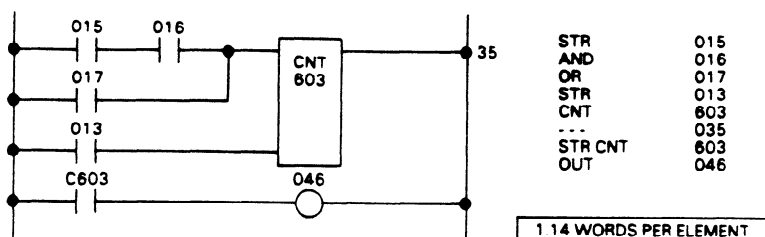


Figure 5-18. Example of Counter Logic

Figure 5.19 illustrates the operation of counters. Again, the relay logic driving the counters is built as separate rungs using normal relay programming techniques. Counter 603 will count (increment) whenever references 015 and 016 are both ON or reference 017 is ON. If one parallel path is energized while the other is also ON, no additional count is recorded. When the count reaches 35 (its preset), coil 603 is energized, also energizing coil 46. Counting will continue beyond the preset value, and will continue counting until the counter is reset. Whenever reference 013 is energized, regardless of the count or top rung status or coil state, counter 603 will be reset to zero and held at that value until reference 013 is de-energized.

pc-s1-83-0018



KEY	ON LED'S	DISPLAY	KEY	ON LED'S	DISPLAY	KEY	ON LED'S	DISPLAY
STR	STR		STR	STR		STR	STR	
SHF	STR, SHF		SHF	STR, SHF		CNT	STR, CNT	
1	STR, SHF	1	1	STR, SHF	1	SHF	STR, CNT, SHF	
5	STR, SHF	15	3	STR, SHF	13	6	STR, CNT, SHF	6
ENT	ADR	0 0 0 1	ENT	ADR	0 0 0 4	0	STR, CNT, SHF	60
AND	AND		CNT	CNT		3	STR, CNT, SHF	603
SHF	AND, SHF		SHF	CNT, SHF		ENT	ADR	0 0 0 7
1	AND, SHF	1	6	CNT, SHF	6	OUT	OUT	
6	AND, SHF	16	0	CNT, SHF	60	SHF	OUT, SHF	
ENT	ADR	0 0 0 2	3	CNT, SHF	603	4	OUT, SHF	4
OR	OR		ENT	ADR	0 0 0 5	6	OUT, SHF	46
SHF	OR, SHF		SHF	SHF		ENT	ADR	0 0 0 8
1	OR, SHF	1	3	SHF	3			
7	OR, SHF	17	5	SHF	35			
ENT	ADR	0 0 0 3	ENT	ADR	0 0 0 6			

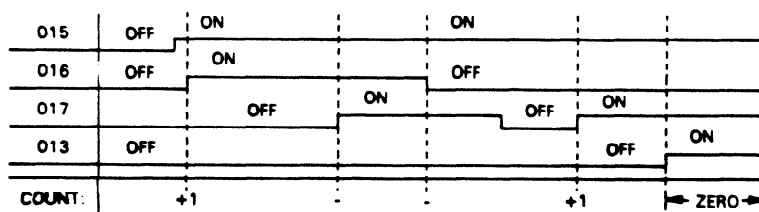


Figure 5-19. Example of Counters

Extending the Timer and Counter Range

Timers and counters can be connected in series or cascaded to extend their range beyond four digits. Figure 5.20 illustrates several techniques to extend preset ranges by utilizing multiple timer/counter

GEK-90842

functions. Timer 611 will record time as soon as references 206 and 225 are energized. After 800 seconds it energizes its coil and stops recording time. However, timer 612 starts as soon as 611 reaches its preset and continues for another 950 seconds. This is a total of 1750 seconds from the time T611 started. Whenever reference 206 or 225 is de-energized, timer 611 is reset to zero, de-energizing coil 611, and also resetting timer 612 (if necessary).

pc-s1-83-0019

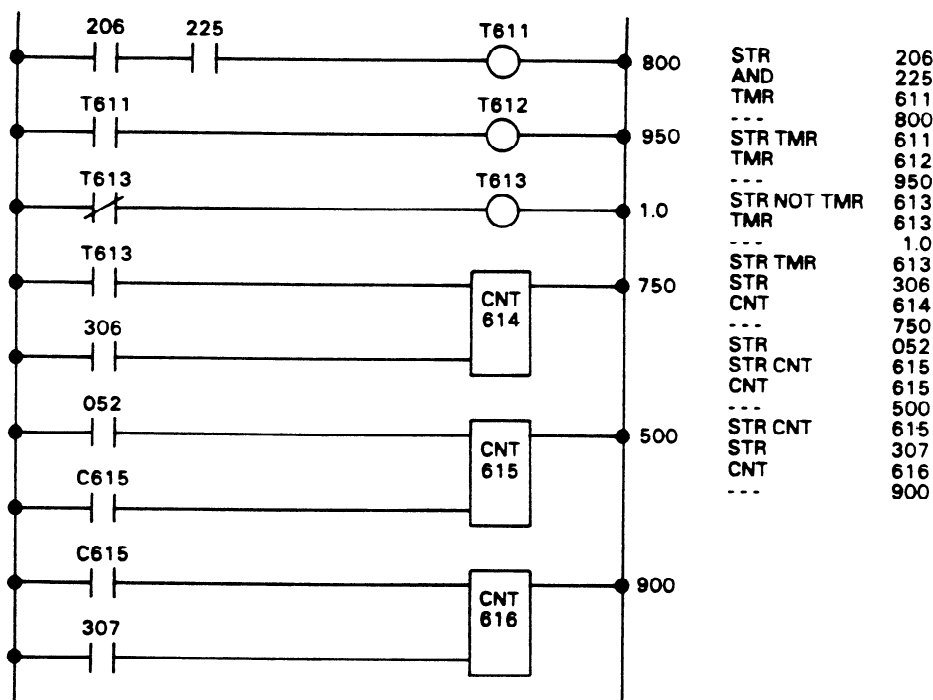


Figure 5-20. Example of Extended Counters

Another pair of functions are timer 613 and counter 614. Timer 613 is a self resetting timer with a preset of one second (10 tenths). It produces outputs each one scan long, every second from the time the CPU starts until it is turned OFF. Counter 614 counts these pulses up to 750 of them (750 seconds) until reset by reference 106. Since counters are retentive, using timer 613 to produce a one second clock that is counted by 614, results in a retentive timer. A similar result occurs if the top rung of counter 614 were referenced to internal coil 375 in lieu of 613. The third example in Figure 5.20 is counters 615 and 616. Counter 615 counts reference 52, up to 500, and then resets itself. Counter 616 records how many of these groups of 500 counts occur. Since counter 616 has a preset of 900, its coil will be energized only after 450,000 transitions of reference 52. At anytime, counter 615 has a representation of the small number of counts (0-499 or least significant portion) and counter 616 the larger values representing multiples of 500 (500-450,XXX) or most significant portion. Application techniques shown in Chapter 8 illustrate how a low order preset of 1000 can be obtained to produce a more conventional double precision counter.

Sequencer Operation

All Series One and Series One Plus counters have one other very powerful feature which is applicable to all counters. They all can be programmed to act as sequencers. Each counter can control a 1000 step sequencer, which allows any Series One or Series One Plus PC to have sixty four 1000 step sequencers, each similar to a stepping switch or drum sequencer. Figure 5.21 illustrates the operation of one of these sequencers. Each sequencer moves from one position to the next as directed by some user defined signal shown here as a pushbutton. Counters move or are incremented once each time a count is recorded. Each sequencer starts at zero or home position when the counter is reset, and progresses through its steps one at a time in numerical order. At any time the position of the sequencer can be determined by the current count value contained in its storage. Although any sequencer can have up to 1000 steps, many applications are solved with a smaller number. To limit a sequencer to less than 1000 steps, the preset on the counter is used. As usual, each step of the sequencer can control any number of contacts.

pc-s1-83-0020

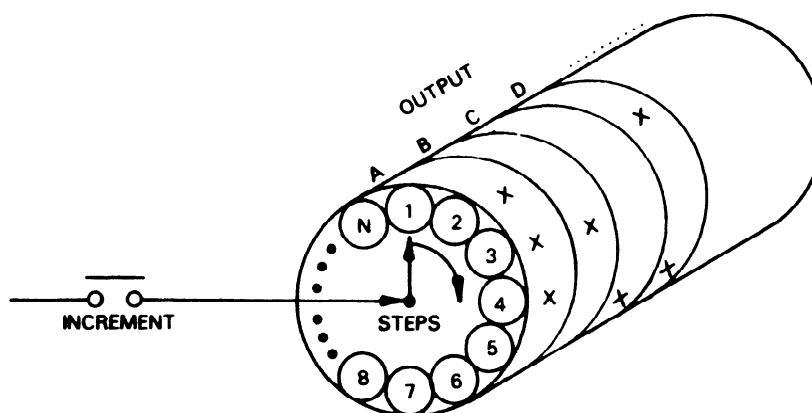


Figure 5-21. Illustration of Sequencer Operation

To illustrate the capabilities of the sequencer, refer to the example in Figure 5.22. The sequence map in this example is in the form of a timing diagram that has an overall cycle of 45 seconds. This time is subdivided into increments of 5 seconds per the requirements of this application. There are six outputs to be controlled during the sequence with the desired ON periods shown by horizontal black lines. Outputs are assigned and step numbers (starting at zero) are entered for each time period. To drive the sequencer (Figure 5.23), a timer (600) is entered to reset itself each 5 seconds. Input 030 is used to control the sequencer, which will not be incremented except when this input is energized. When energized, timer 600 will produce one pulse every 5 seconds. Counter 601 increments its count once each pulse, thus stepping the sequencer. Input 031 will reset the sequencer to home (zero) whenever it is energized, regardless of the count or position of the sequencer. The counter will go up to eight and then stop waiting for a reset signal.

GEK-90842

pc-s1-83-0021

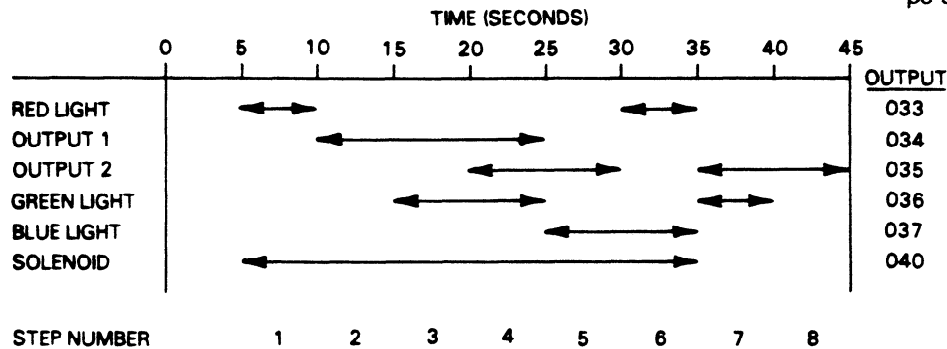


Figure 5-22. Example of Sequencer

The outputs from the sequencer are programmed using relay logic. However, the reference is first to the counter (601) controlling the sequencer and then to the current value that is going to control that contact. For example if a reference to counter 601 is desired that will pass power only when that counter has a current count of exactly one, the value 1 is entered after the reference to 601 on a normally open contact.

In this example, output 033 (red light) will be energized during 5 second intervals each for steps 0, 1, and 6. Similarly, output 034 (Output 1) will be energized at step 2, sealed and held until the beginning of step 5 (end of step 4). The other outputs are controlled using similar logic with parallel contacts or seal-in circuits.

If an output is ON for more steps than it is OFF, normally closed contacts would probably simplify your logic. Counters can also be used that are driven independently and not with a time base.

Referencing Sequencer Contacts with a Series One Plus PC

When programming a Counter or Shift Register in a Series One Plus or Series One Plus 3.7K PC that has the IC610CPU105 CPU installed, a Drum Sequencer contact reference should not be used as the first contact (with a STR or STR NOT instruction) in the COUNT or RESET rung of a counter or the DATA, CLOCK, or RESET rung of a Shift Register. If a program using the step described above was entered into a Series One PC having an IC610CPU101 CPU (this would work correctly) and written to Tape, then later transferred to a Series One Plus PC with an IC610CPU105 CPU or a Series One Plus 3.7K PC with an IC610CPU106 CPU, the program would not execute properly.

Do not reference a Drum Sequencer step as the first contact in the rungs as described above. If a Drum Sequencer step is required as the first contact reference for a rung in a Counter or Shift Register, a programming technique that would work properly is to use the required Drum Sequencer step to turn on an internal coil in a separate rung, then reference that coil as the first contact in the Counter or Shift Register rung.

GEK-90842

Printer Interface Unit
CROSS REFERENCE PRINTOUT

OUTPUTS USED TABLE V2. 2

SERIES ONE, ONE PLUS/SR-20, SR-21

PAGE 0007

SHIFT REGISTER

400:	410: *	420: *	430: *	440:	450:	460:	470:
401:	411: *	421: *	431: *	441:	451:	461:	471:
402:	412: *	422: *	432: *	442:	452:	462:	472:
403:	413: *	423: *	433: *	443:	453:	463:	473:
404:	414: *	424: *	434: *	444:	454:	464:	474:
405:	415: *	425: *	435: *	445:	455:	465:	475:
406:	416: *	426: *	436: *	446:	456:	466:	476:
407:	417: *	427: *	437:	447:	457:	467:	477:

500:	510:	520:	530:	540:	550:	560:	570:
501:	511:	521:	531:	541:	551:	561:	571:
502:	512:	522:	532:	542:	552:	562:	572:
503:	513:	523:	533:	543:	553:	563:	573:
504:	514:	524:	534:	544:	554:	564:	574:
505:	515:	525:	535:	545:	555:	565:	575:
506:	516:	526:	536:	546:	556:	566:	576:
507:	517:	527:	537:	547:	557:	567:	577:

TIMER/COUNTER

600:	610:	620:	630:	640:	650:	660:	670:	100:
601: *	611:	621:	631:	641:	651:	661:	671:	101:
602:	612:	622:	632:	642:	652:	662:	672:	102:
603: *	613:	623:	633:	643:	653:	663:	673:	103:
604:	614:	624:	634:	644:	654:	664:	674:	104:
605:	615:	625:	635:	645:	655:	665:	675:	105:
606:	616:	626:	636:	646:	656:	666:	676:	106:
607:	617:	627:	637:	647:	657:	667:	677:	107:

Figure 4-8. Sample Outputs Used Table Printout

Printer Interface Unit
CROSS REFERENCE PRINTOUT

OUTPUT REGISTERS USED TABLE V2. 2

SERIES ONE PLUS/SR-21

PAGE 0000

OUTPUT

000:	001:	002:	003:	004:	005:	006:	007:	010:	011:	012:
013:	014:	015:		070:	071:	072:	073:	074:	075:	076:

INTERNAL RELAY

016:	017:	020:	021:	022:	023:	024:	025:	026:	027:
030:	031:	032:	033:	034:	035:	036:	037:		

SHIFT REGISTER

040:	041:	042:	043:	044:	045:	046:	047:
050:	051:	052:	053:	054:	055:	056:	057:

DATA REGISTER

400:	410:	420:	430:	440:	450:	460:	470:
401:	411:	421:	431:	441:	451:	461:	471:
402:	412:	422:	432:	442:	452:	462:	472:
403:	413:	423:	433:	443:	453:	463:	473:
404:	414:	424:	434:	444:	454:	464:	474:
405:	415:	425:	435:	445:	455:	465:	475:
406:	416:	426:	436:	446:	456:	466:	476:
407:	417:	427:	437:	447:	457:	467:	477:
500:	510: *	520:	530:	540:	550:	560:	570:
501:	511: *	521:	531:	541:	551:	561:	571:
502:	512:	522:	532:	542:	552:	562:	572:
503:	513:	523:	533:	543:	553:	563:	573:
504:	514:	524:	534:	544:	554:	564:	574:
505:	515:	525:	535:	545:	555:	565:	575:
506:	516:	526:	536:	546:	556:	566:	576:
507:	517:	527:	537:	547:	557:	567:	577:

Figure 4-9. Sample Outputs Used Table Printout (continued)

pc-s1-83-0022

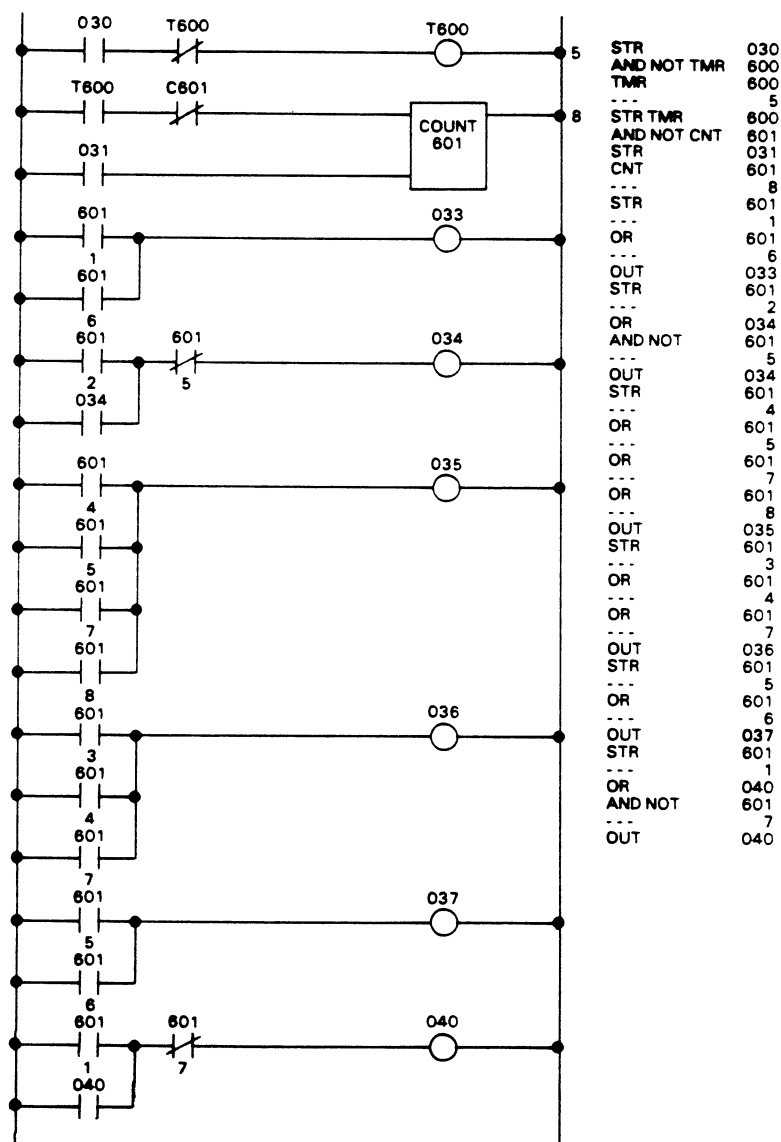


Figure 5-23. Typical Sequencer Logic

Shift Register Functional Description

Another powerful feature of all Series One and Series One Plus PCs is the ability to simulate the operation of shift registers. There are many physical devices that operate similar to shift registers such as an anchor chain, a conveyor belt, an indexing machine, a line of customers at a refund desk, etc.

GEK-90842

There are several common features that can be used to describe the generalized features of a shift register. There is movement in all examples, normally in one direction. This movement can be defined as a group of fixed increments such as one link, one box position, one operation, or one person. The 2 order 1 of activities (pulling chain in, placing boxes on conveyor, loading parts to be worked on/tested, or adding people to the line) normally remains the same - first in, first out. In case of power failure in the plant or building, it is desirable that the shift register retain its content (part position, people order, boxes on conveyor, etc). However, in many cases it is also desirable that under logic control, the shift register be cleared such as start up on Monday morning. It is much easier for the user to clear a retentive shift register than it is to restore a cleared (volatile) content.

pc-s1-83-0023

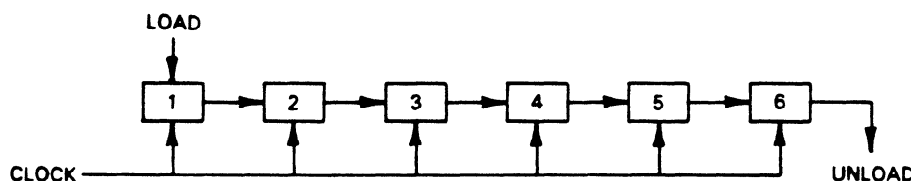


Figure 5-24. Typical Shift Register

A shift register is a group of storage locations that are synchronized by a timing or clock signal. Figure 5.24 illustrates the operation of a typical shift register with six stages. Each stage can store a single bit which can represent a good or bad part, part or no part, ON or OFF, a one or zero, etc. Data is loaded into the first stage, incremented through the shift register one position per clock pulse, and then out the other end. When the clock signal goes from OFF to ON, each stage shifts one position towards the right. The content of stage 6 is unloaded, stage 5 content moves into stage 6, stage 4 into 5, etc. until stage 1 moves into stage 2. Stage 1 is then cleared, awaiting a load signal from another logic.

Within the Series One and Series One Plus, there are 128 retentive storage locations for use with shift registers. Since the Series One family logic functions are easy to use yet provide the user with unmatched flexibility, they allow many separate shift registers to be built of various lengths as long as the total quantity of stages does not exceed 128.

Each shift register requires several pieces of information to perform its function. Since it is a serial operation, there must be a definite beginning and a similar ending. At the start of an operation parts are put into the machine or boxes are put on the conveyor. A signal is required to indicate when a part is to be placed into the shift register at stage one. Similarly, there is a signal that indicates when the shift register should increment or move one position. The third and last signal is one to indicate when the shift register (all stages) should be cleared to zero or OFF conditions.

Shift Register References

The definition of size for any shift register requires the user to define how many stages are desired, and which references are assigned to this function. The size of the shift register is usually established by the maximum size of the physical device it is controlling. For example, how many links are in the chain, how many boxes can be placed on the conveyor, how many positions has the index machine, or how many people is there space for? The references to stages of the shift registers within the Series One and Series One Plus are to the references 400 through 577. Each stage or position of any shift register should be unique. Storage locations should not be shared between different shift registers. However, multiple logic functions can be programmed to operate upon one shift register to produce bi-directional

shift registers, ring counters, First In First Out or Last In First Out stacks. Each shift register reference can control any number of relay contacts, both normally open and normally closed.

Shift Register Operation

To illustrate the operation of serial shift registers, Figure 5.25 defines a theoretical problem using a sequential assembly machine. The machine has space for a maximum of 12 parts at any one time. These parts may be cups or other storage locations physically on the machine. The operations are performed while the machine is stationary and then all storage locations (cups) are indexed rapidly one position towards the right. There are three separate operations such as part assembly, welding, clamping, nuts tightened, painting and label placement, etc. Since it would be a waste of material and machine time to operate on locations without a part in place, there is a sensor on the first location to detect when a part is present. To the right is a testing position to determine good parts and bad parts for later separation. Each position of the machine is assigned a number for later reference to the shift register. Notice that positions 3, 5, 7, 10, and 12 are only holding locations to store parts in progress and are related to the mechanical layout of the machine.

pc-s1-83-0024

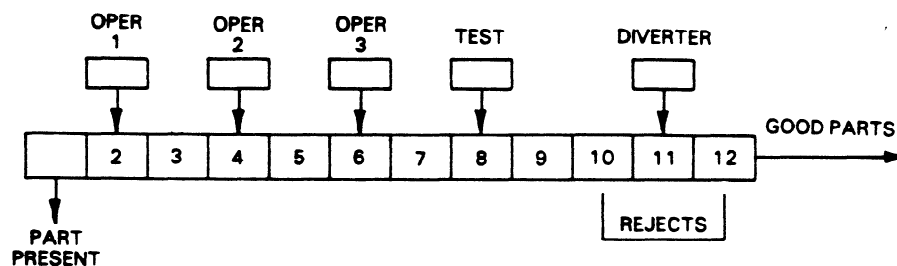


Figure 5-25. Example of Shift Register

Figure 5.26 illustrates the logic, including shift register, used to solve this problem. The following I/O references are assigned for use in this logic and can be changed as necessary to fit your application.

Part Present (Input)	045
Test OK (Input)	046
Reset/All (Input)	047
Operation 1 (Output)	130
Operation 2 (Output)	131
Operation 3 (Output)	132
Test (Output)	133
Diverter (Output)	134
Shift Permissive (Internal)	214

The timing of each operation allows the machine to index one position each 3 seconds. Thus timer 631 produces an output every 3 seconds as long as the internal permissive is satisfied. This coil (214) is developed elsewhere and indicates all operations have been completed and hardware is out of the way, allowing movement without damaging the machinery. The shift register itself has three parallel rungs each of which can be any combination of series and parallel contacts. The first is the input signal. Whenever this is supplying power flow the first stage of the shift register will turn ON and remains ON

GEK-90842

even if the input changes. The middle rung is the clock signal controlling the shifting of all stages. When this signal goes from OFF (no power flow) to ON (power flow), all stages will shift one position. This shifting is completed before other logic is solved. The bottom rung is the reset signal. Whenever this signal is supplying power flow, (all) stages of the shift register will be turned OFF (cleared).

The references used by this shift register and thus the number of stages, must be defined. The reference for the shift register itself is the location of the first stage and immediately following that is the reference for the last stage of that shift register. In our example, input 045 indicates a part present and loads the first stage (431) of the shift register.

Timer 631 (3 seconds) and coil 214 (permissive to shift) are both required before the shift occurs. Finally, input 047 is used to clear all shift register stages perhaps at the start of machine operation or otherwise under operator control. The shift register is built using references 431 through 444 (total 12 stages). The remaining relay logic merely connects the individual stages of the shift register to their output to exercise proper control. The only exception is the next to the last rung. This rung receives the test signal and resets stage 440 if the test is passed. Thus when the later stage (443) receives the data, it will not activate coil 134 to operate the diverter. Therefore, good parts are passed and the diverter operates only for bad parts, not for empty locations or good parts.

a42154

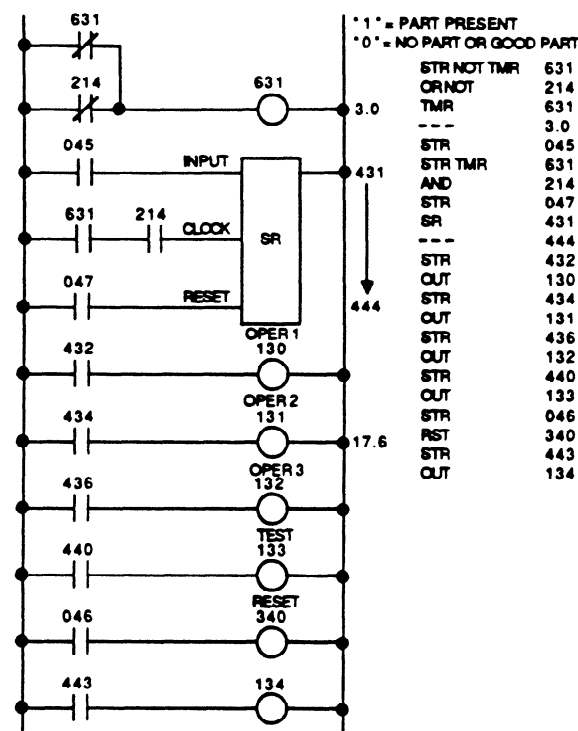


Figure 5-26. Example of Shift Register Logic

Normally many parts will be processed through this machine, continuously moving from input at position 1 to output from position 12. However, for the purpose of illustration, a single part will be examined as it moves down the machine. The part is placed in position 1, energizing input 045 and setting shift register stage 431. After three seconds, timer 631 energizes clocking the shift register causing the ON state for this part to move into stage 432. Then timer 631 will reset itself and stage 431 is cleared. When stage 432 is energized so is output 130 causing operation one to be performed on this part. An additional three seconds pass and another clock cycle causes the ON to move to stage 433. Stage 432 will be loaded from 431 which is OFF as long as additional parts are not processed. The above discussion assumes no delays are experienced with the permissive.

The above operation repeats itself as the part goes through 434 (operation 2), 435 (rest), 436 (operation 3), 437 (rest), and 440 (test) at three second intervals. While at the test position, output 133 is energized and the test OK input (046) is expected. If the part passes the test, input 046 will reset stage 440 making the remainder of the shift register act as if no part was present and allowing the part to pass straight on through. However, if the test is not passed, the ON stage moves to 441, 442, and 443 at three second intervals. At stage 443, it energizes output 134 causing the diverter to push off the defective part into the reject bin.

Other logic can be added to improve the capabilities of this program. For example, a timer could measure the delay from the manual cycle (631 energized) and receipt of the permissive (214). It would also alarm if this delay was excessive such as 5 or 8 seconds beyond the normal 3 seconds. Counters could also be installed to record both good and bad parts produced. This data can be readout and reset via the programmer whenever desired, such as each shift, day, week, etc.

Forcing I/O References

An invaluable tool in the verification of user logic and field wiring is the ability to force I/O references. The operation of the I/O forcing operation is dependent on the I/O reference being forced. Inputs connected to external devices can be forced on or off for one solution of user logic. All other I/O references can be forced on or off indefinitely, but are overridden by user logic.

The most frequent user of the I/O forcing function is for verification of field wiring of outputs. Since user logic overrides the I/O forcing function it is recommended that this operation be performed with user program memory cleared. The key sequences for forcing I/O references are as follows:

- To force a specific I/O reference ON, enter the sequence: SET, SHF, XXX (I/O reference), ENT.
- To force a specific I/O reference OFF, enter the sequence: RST, SHF, XXX (I/O reference), ENT.

WARNING

When forcing input points with the set or RST sequence, be aware that the physical state of the input may be overridden. If the forced set or RST occurs in the user logic program before the input is checked in the same I/O scan, the set or RST state will take precedence and would cause an output to be turned on or off at the wrong time.

SECTION 3

Data Operation Instructions for the Series One Plus Programmable Controller

Data Operations

The Series One Plus PC has, in addition to the basic ladder diagram functions, a group of instructions that include data moves, math functions, logical operations, conversion, and external fault diagnosis. These instructions provide the PC with the capability of performing various data operations. This group of instructions operate on multiple bits rather than one bit at a time. These instructions are not available with a Series One PC. For programs requiring many data operations, it is recommended that the Series One Plus 3.7K PC be used.

All of these instructions require that the data to be operated on be loaded (stored) into a 16-bit register called the accumulator. The data (numerical value) is processed in the CPU as BCD (Binary Coded Decimal). Each BCD value is represented by 4 corresponding bits in the accumulator. A BCD value can be any of the digits 0 through 9. When the specified operation is performed with the contents of the accumulator, the result of the operation is stored in the accumulator. This data can then be transferred to external outputs or stored internally to be used as needed. Additionally, data can be stored in registers (sixty-four 16-bit registers) and those registers referenced when programming the data operations. Each register can contain a 4-digit BCD number. Figure 5.27 is a block diagram showing how the data operations are performed.

Registers and the accumulator may also contain binary data, however arithmetic functions cannot be performed until the binary values are converted to BCD.

a40848

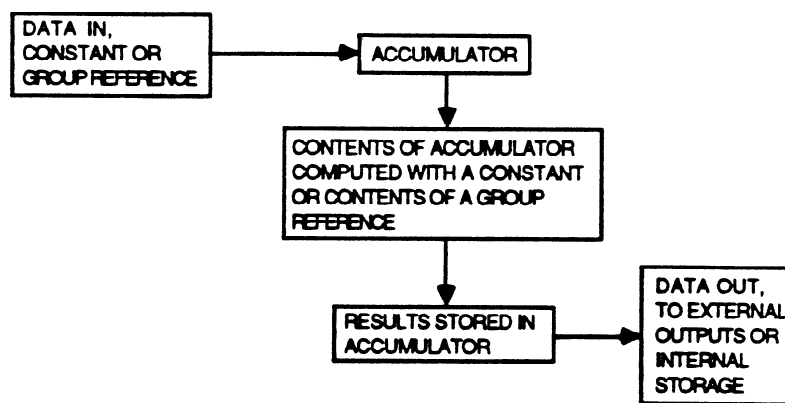


Figure 5-27. Data Operation Block Diagram

The contents of the accumulator do not change until new data is loaded into it with a D STR instruction or a computation takes place and the result is stored in the accumulator.

Using References for Data Operations

When entering an instruction for a data operation, the instruction is first specified, then the data. The data can be a constant value or a group reference that contains the data to be used.

Entering a Constant Value

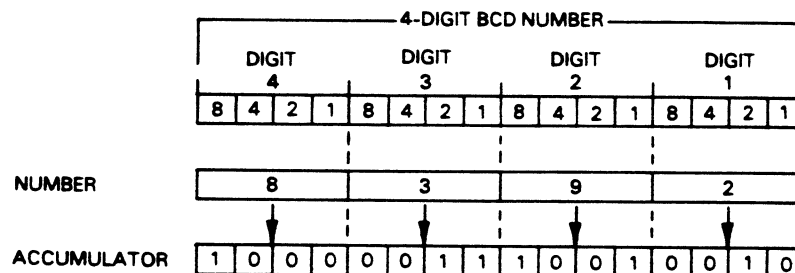
a40849



Where XXXX is a 4-digit BCD number

When a constant (4-digit BCD number) is entered as the data value, the bits of the BCD numbers will be stored in the corresponding position in the accumulator, i.e., the least significant digit in the lower 4 bits of the accumulator, and the most significant digit in the upper 4 bits of the accumulator.

a40850



Entering a Group Reference



The I/O points, internal coils, and shift registers are treated as 8-bit registers when programming data operations. The references are then referred to as group references. By selecting the R key before a value, that value is operated on as an 8-bit or 16-bit register. For example, R 1 5 actually references 8 consecutive I/O points, 150 to 157. This is a convenient way of reading in data to the accumulator from input modules, or writing data from the accumulator to output modules for use as required by the user program. Certain instructions operate on 16 bits, in those cases the data will be read from or written to 16 consecutive references (one 16 circuit module or 2 adjacent 8 circuit modules). The group reference number for the references mentioned above can be either 1 or 2 digits.

GEK-90842

Data Register References

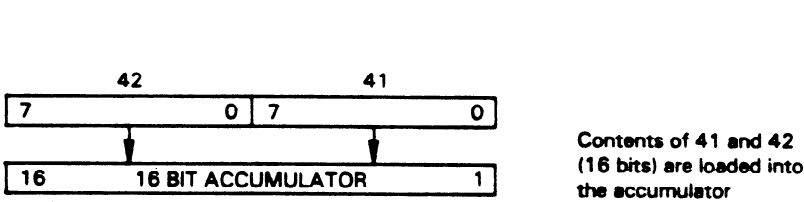
Data registers are referenced as 3-digit octal values with a valid range from 400 to 577. Although each individual register actually refers to one 8-bit byte, most registers are operated on as 16-bit words (2 consecutive registers). For example, if the key sequence R 4 0 2 is entered, the data operation uses data registers 402 and 403. If a 4-digit BCD value (for example 1234) were entered into the register, register 402 would contain the 2 least significant digits (34) and 403 the 2 most significant digits (12). A useful feature of the registers is that they can contain a value to be used as the preset for timers or counters. Unused T/C references 600-673 can be used as data registers. Each T/C reference operates on a 16-bit register.

Timer/Counter References

The timer/counter references are also 3-digit octal values. Each of the timer/counter references represent a 16-bit register. Valid range for the timer/counter references is 600 to 677. These registers contain the accumulated value of a timer or counter. It is important to note that the timer/counter accumulated value is not the same as the accumulator read to or written from when programming any of the data operations.

Example of Specifying a Group Reference

An example of specifying a group reference and how it relates to the accumulator is shown below. Example: Reference specified is 41.



NOTE

Discrete bit 410 will be stored in the first position of the accumulator; discrete bit 427 will be stored in the last position (16) of the accumulator.

Programming the Data Operation Instructions

When a program is entered to perform any of the data operations, the first instruction must have at least one conditional contact entered immediately before it. To enter any of the data instructions, the F key must first be selected, then the Function number (F50, F63, F82, etc.) of the instruction. These numbers are assigned internally, by the CPU's operating system, to each of the data operation instructions and are listed in Table 5.8. After the F X X number has been entered, the program address will advance 1 step and the second word (data entry) should now be entered. The data entry is selected by entering SHF X X X X, for a constant value, or R X X X for a group reference.

Table 5-8. Data Operation Function Numbers

Instruction	Key Sequence	Instruction	Key Sequence
D•STR	F50	MULTIPLY	F73
D•STR1	F51	DIVIDE	F74
D•STR2	F52	D•AND	F75
D•STR3	F53	D•OR	F76
D•STR5	F55	SHIFT RIGHT	F80
D•OUT	F60	SHIFT LEFT	F81
D•OUT1	F61	DECODE	F82
D•OUT2	F62	ENCODE	F83
D•OUT3	F63	INV	F84
D•OUT5	F65	BIN (BCD TO BIN)	F85
CMPR	F70	BCD (BIB TO BCD)	F86
ADDITION	F71	EXTERNAL FAULT	
SUBTRACTION	F72	DIAGNOSIS	F20

The remainder of this chapter lists each of the data operation instructions and the information required for programming them. The format used to describe each of the instructions is as follows:

Mnemonic name for each instruction.

Function number that must be entered with the programmer (F50, F60, etc.)

Illustration of how each instruction operates in relation to the accumulator.

List of valid references (where required).

Keystrokes required to enter the instruction.

Program example to enhance the description of certain instructions.

Special Function Coils for Data Operations

A group of 6 special purpose internal coils in the Series One Plus are used as indicators of certain conditions that may occur when some of the data operations are performed. When these conditions occur, the applicable coil is turned on as a "Flag" or indicator that the condition has occurred. These coils can be included in your program as conditional contacts to turn on an output or perform some function to indicate that a Flag is on. These special purpose coils and their meanings are:

COIL REFERENCE	FLAG INDICATION
772	>, Greater Than
773	=, Equal to
774	<, Less Than
775	Carry or Borrow
776	Zero
777	Overflow

GEK-90842

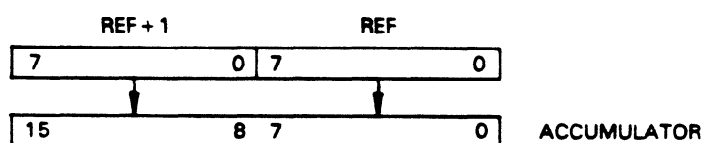
D•STR (F50)

D•STR (Data Store) is a 2 word instruction which, when executed, loads the accumulator with a 4 digit BCD constant or the contents of a specified 2 byte reference.

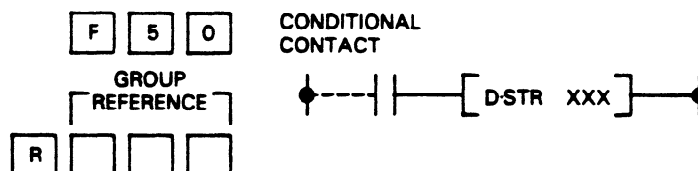
NOTE

When loading I/O References or other data, be sure that it is in BCD Format. If not it should be converted before using the data for arithmetic functions.

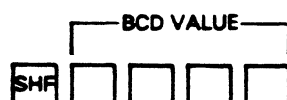
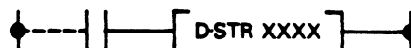
a40852



Valid Group References	
000 - 014	I/O Points
070 - 075	I/O Points
016 - 036	Internal Coils
040 - 056	Shift Register Coils
400 - 576	Data Registers (8-bit)
600 - 677	T/C Accumulated Value Registers (16-bit)
0000 - 9999	Constant (4-digit BCD)



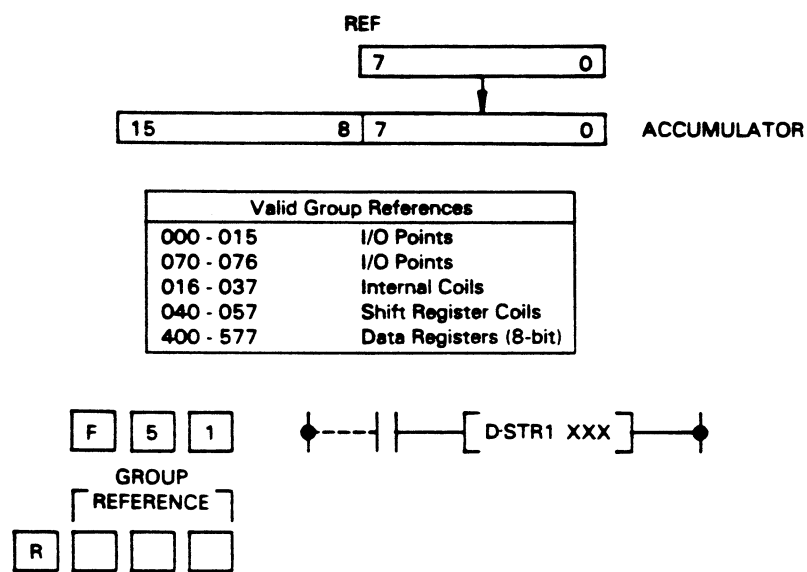
4 DIGIT CONSTANT



D•STR1 (F51)

This is a 2 word instruction which, when executed loads the lower 8 bits of the accumulator with the contents of a specified 1 byte reference. The upper 8 bits (8-15) will be zeros.

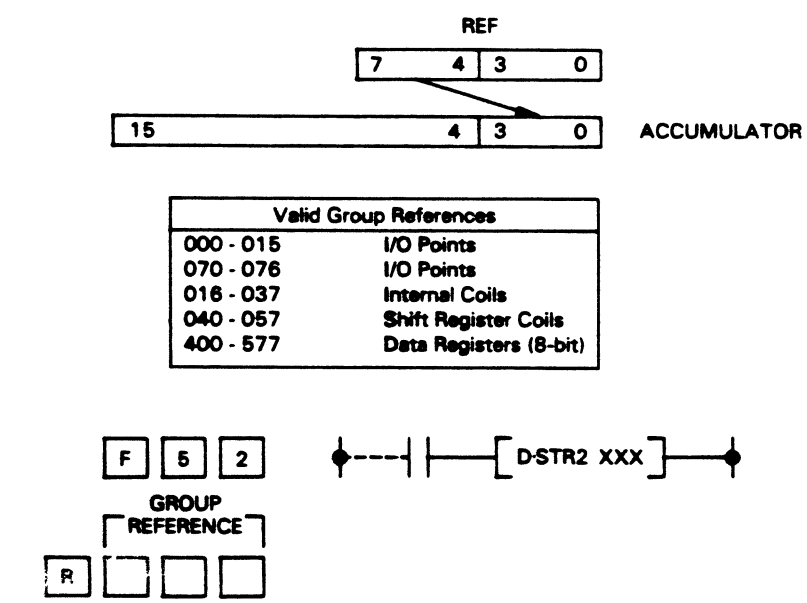
a40853



D•STR2 (F52)

This is a 2 word instruction which, when executed loads the lower 2 bits of the accumulator with the upper 4 bits of a specified 1 byte reference.

a40854

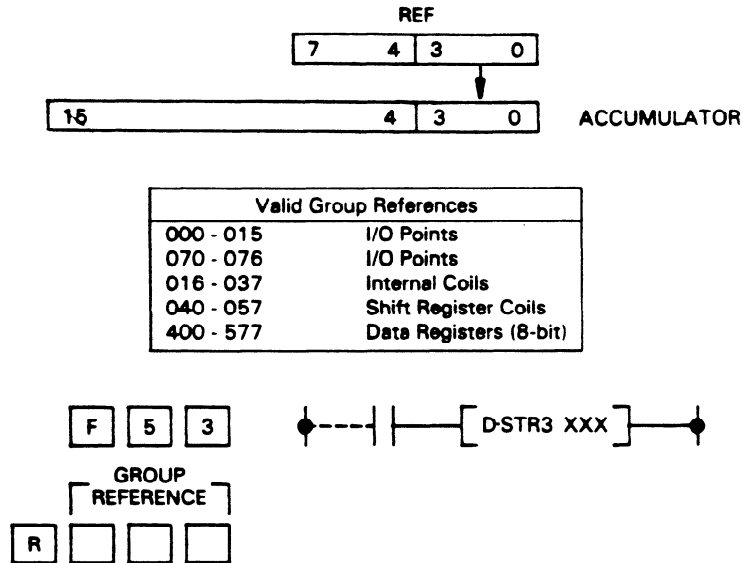


GEK-90842

D•STR3 (F53)

This is a 2 word instruction which, when executed loads the lower 4 bits of the accumulator with the lower 4 bits of a specified 1 byte reference.

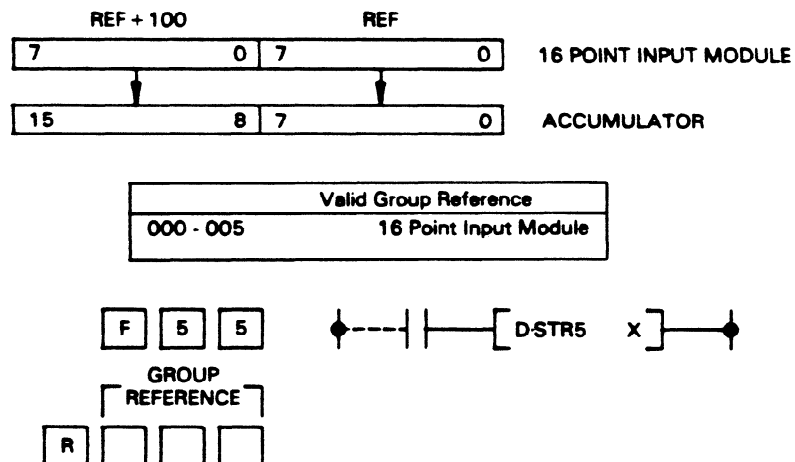
a40855



D•STR5 (F55)

This is a 2 word instruction which, when executed loads the accumulator with a BCD value from the 2 byte contents of a 16 circuit Input module.

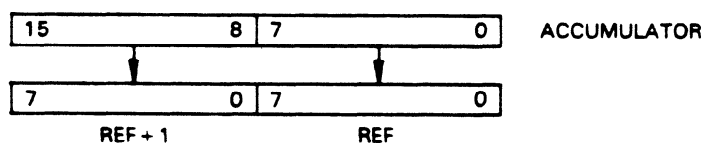
a40856



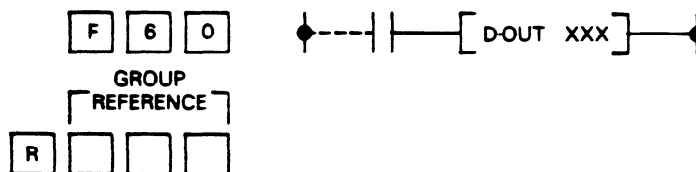
D•OUT (F60)

D•OUT (Data Out) is a 2 word instruction, which when executed transfers the contents of the accumulator (16 bits) to a specified 2 byte (group) reference (Output, Internal Relay, Shift Register coils, Data Registers or Timer/Counter accumulate value register).

a40857

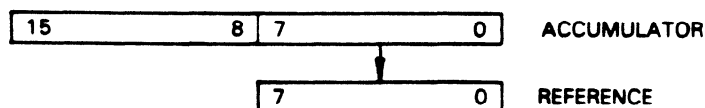


Valid Group References	
000 - 014	I/O Points
070 - 075	I/O Points
016 - 036	Internal Coils
040 - 056	Shift Register Coils
400 - 576	Data Registers (8-bit)
600 - 677	T/C Accumulated Value Registers (16-bit)

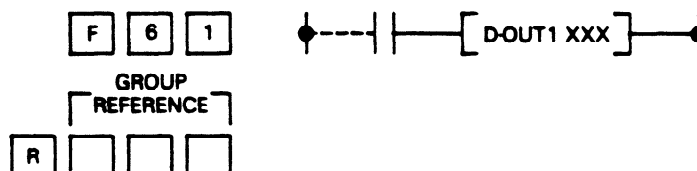
**D•OUT1 (F61)**

D•OUT1 is a 2 word instruction which, when executed will write the contents of the first 8 bits (0-7) of the accumulator to a specified reference.

a40858



Valid Group References	
000 - 015	I/O Points
070 - 076	I/O Points
016 - 037	Internal Coils
040 - 057	Shift Register Coils
400 - 577	Data Registers (8-bit)

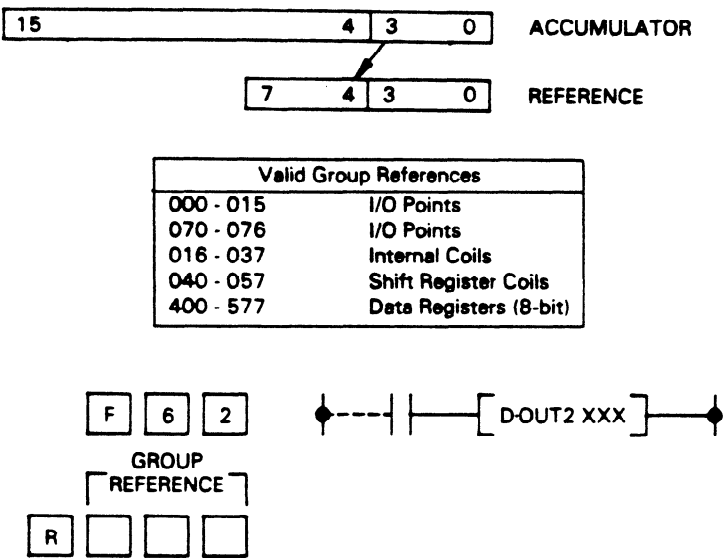


GEK-90842

D•OUT2 (F62)

D•OUT2 is a 2 word instruction which, when executed will write the contents of the lower (first) 4 bits of the accumulator to the upper 4 bits of a specified reference.

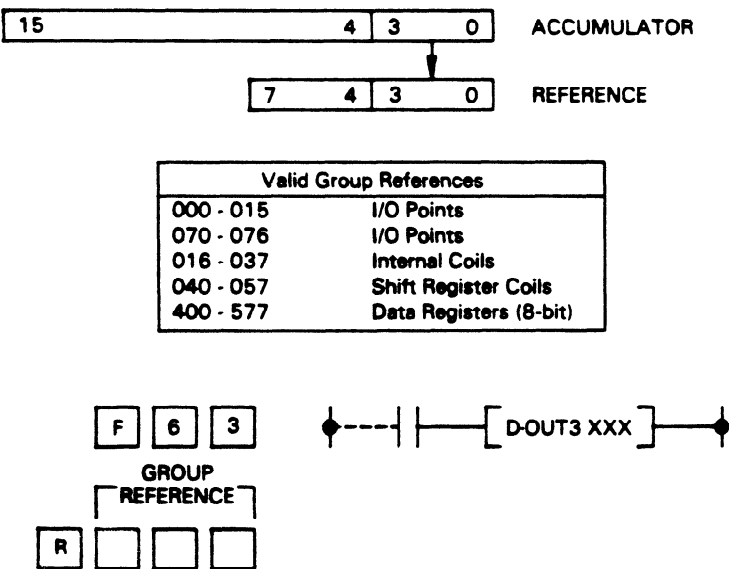
a40859



D•OUT3 (F63)

D•OUT3 is a 2 word instruction, which when executed will write the contents of the lower 4 bits of the accumulator to the lower 4 bits of a specified reference.

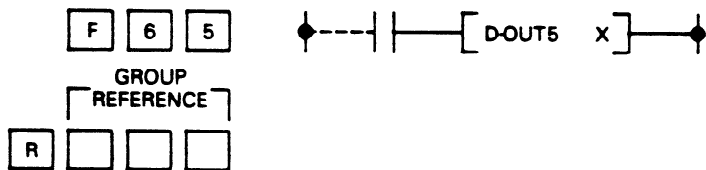
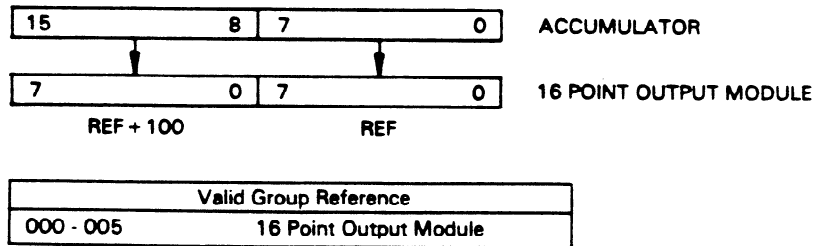
a40860



D•OUT5 (F65)

D•OUT5 is a 2 word instruction, which when executed will write the contents of the accumulator to a 2 byte reference which must be a 16 circuit Output module.

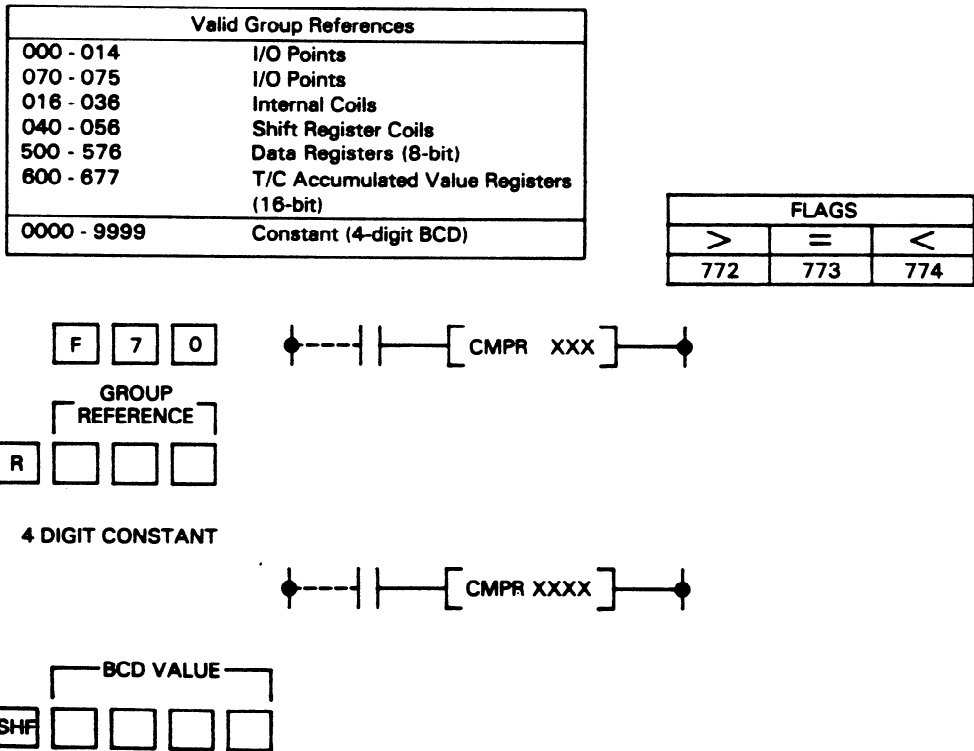
a40861



CMPR (F70)

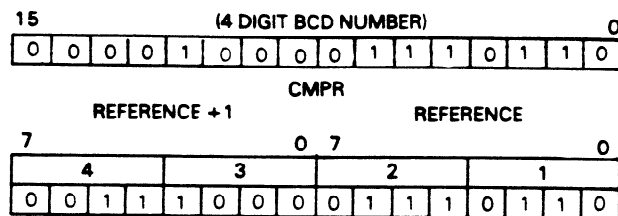
The contents of a specified 2 group reference or a 4 digit BCD constant are compared to the contents of the accumulator. The result of the comparison will cause one of three special purpose internal coils to turn on, depending on whether the accumulator value is greater than (>), equal to (=) or less than (<) the value specified by the reference or constant.

a40862



GEK-90842

• Compare sequence



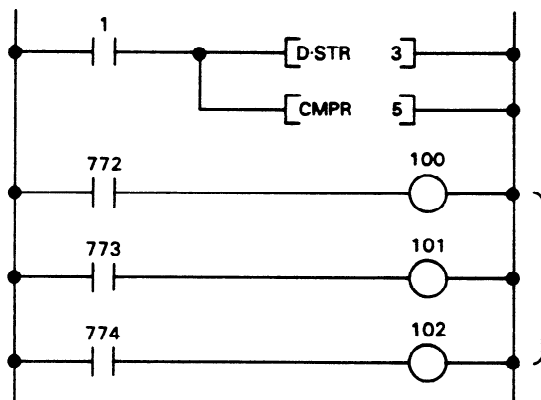
a40863

ACCUMULATOR
CONTENTS

2 GROUP REFERENCE
OR
4 DIGIT CONSTANT

- If Accumulator > Reference, then 772 turns ON.
- If Accumulator = Reference, then 773 turns ON.
- If Accumulator < Reference, then 774 turns ON.

• Programming example, Compare



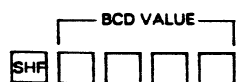
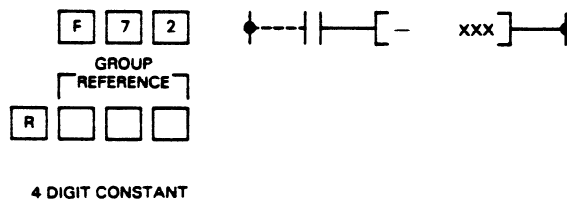
Addition (+), BCD 4 Digit (F71)

The contents of a 4 digit constant or a specified 2 group reference are added to the contents of the accumulator. The result of the addition is stored in the accumulator as a 4 digit BCD number. If the answer is >9999, the carry flag (Internal coil, 775) will turn on. If the answer is zero, the zero flag (776) will turn on. The flag coils can be used in a program as references for contacts to turn on an output connected to a light or other device to provide an indicator of the flag conditions.

a40864

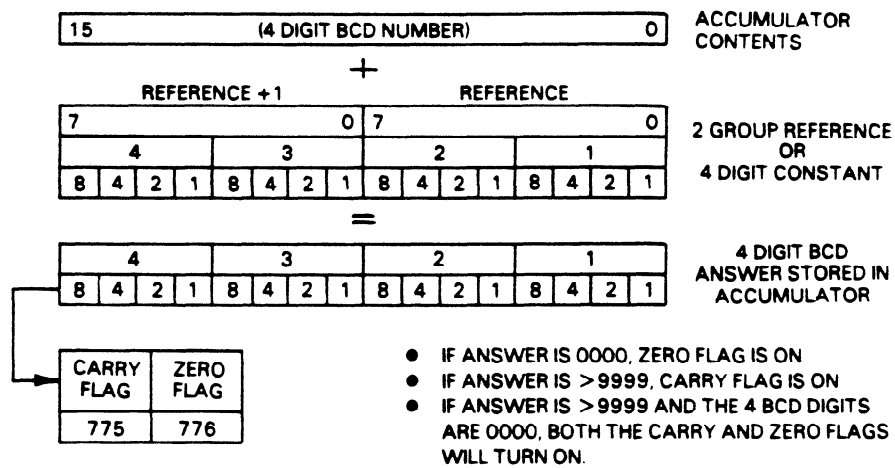
Valid Group References	
000 - 014	I/O Points
070 - 075	I/O Points
016 - 036	Internal Coils
040 - 056	Shift Register Coils
400 - 576	Data Registers (8-bit)
600 - 677	T/C Accumulated Value Registers (16-bit)
0000 - 9999	Constant (4-digit BCD)

FLAGS	
CARRY OR BORROW	ZERO
775	776

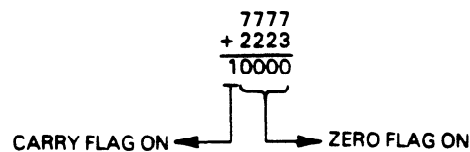


• Addition sequence

a40865



EXAMPLE:

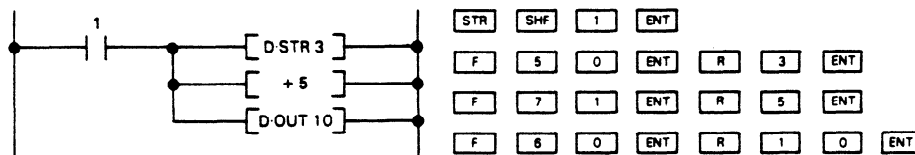


• Addition Programming Example

Example 1

4 BCD digits are loaded in accumulator from Inputs 0030-0047.

a40866



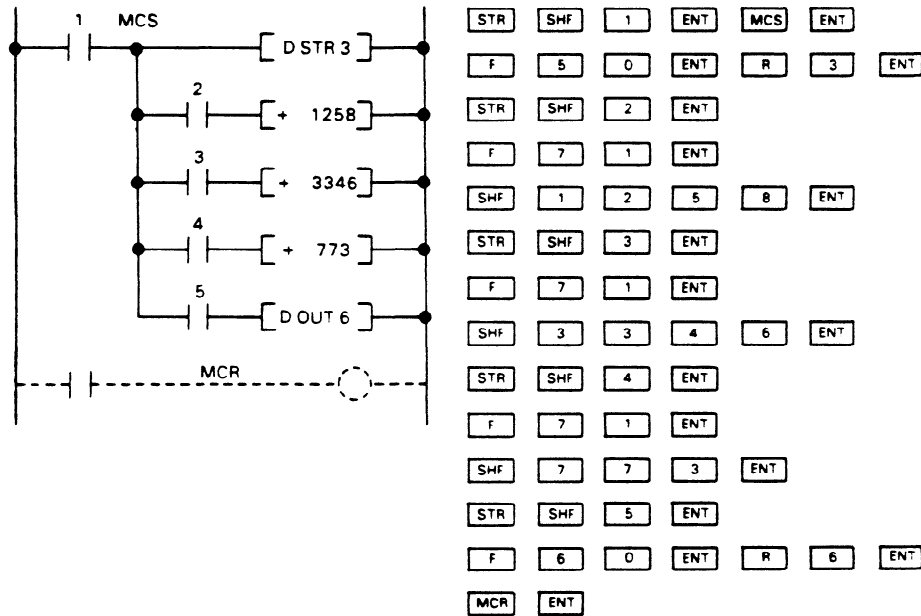
- The contents of the accumulator are added to data from Inputs 0050-0067.
- The answer is stored in the accumulator and also written to Outputs 0100 - 0117. If D OUT is specified as a Data Register, Internal coil, etc.; the answer would be stored at that location for further use in the program.

GEK-90842

Example 2

This example shows more than one constant value entered in the program. By closing the respective contact, a different number can be selected to be added to the contents of the accumulator.

a40867



When I1 is closed, MCS is energized and does not control the logic following it. The number selected by closing one of the contacts, I2, I3, or I4 is added to the contents of the accumulator (read from Inputs 0030-0047). When I5 is closed, the answer will be written to Outputs 0060-0077.

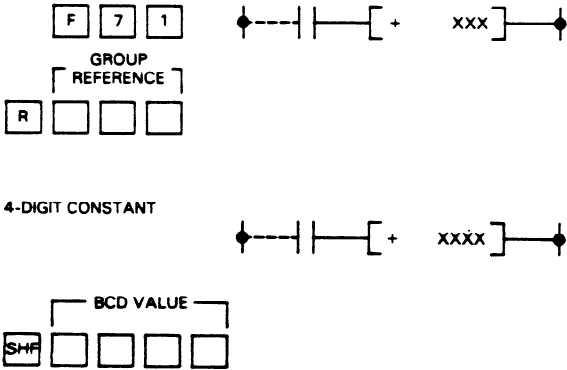
Subtraction (-), BCD 4 Digit (F72)

The contents of a specified 2 group reference or a 4 digit BCD constant value are subtracted from the contents (BCD) of the accumulator. If the answer after the subtraction is positive, it is written directly to the accumulator. If the answer is negative, the Borrow Flag (775) turns ON, the absolute value is subtracted from 10000 and the resulting value is written to the accumulator. If the answer after subtraction is zero, the Zero flag (776) turns ON.

a40868

Valid Group References	
000 - 014	I/O Points
070 - 075	I/O Points
016 - 036	Internal Coils
040 - 056	Shift Register Coils
400 - 576	Data Registers (8-bit)
600 - 677	T/C Accumulated Value Registers (16-bit)
0000 - 9999	Constant (4-digit BCD)

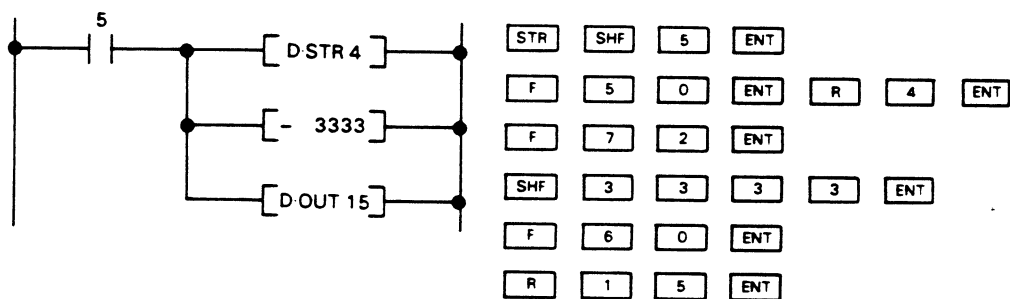
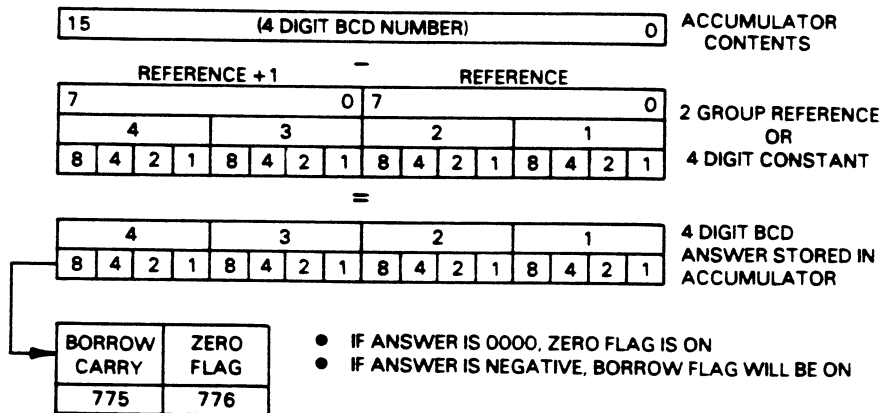
FLAGS	
CARRY	ZERO
775	776



GEK-90842

• Subtraction sequence

a40869



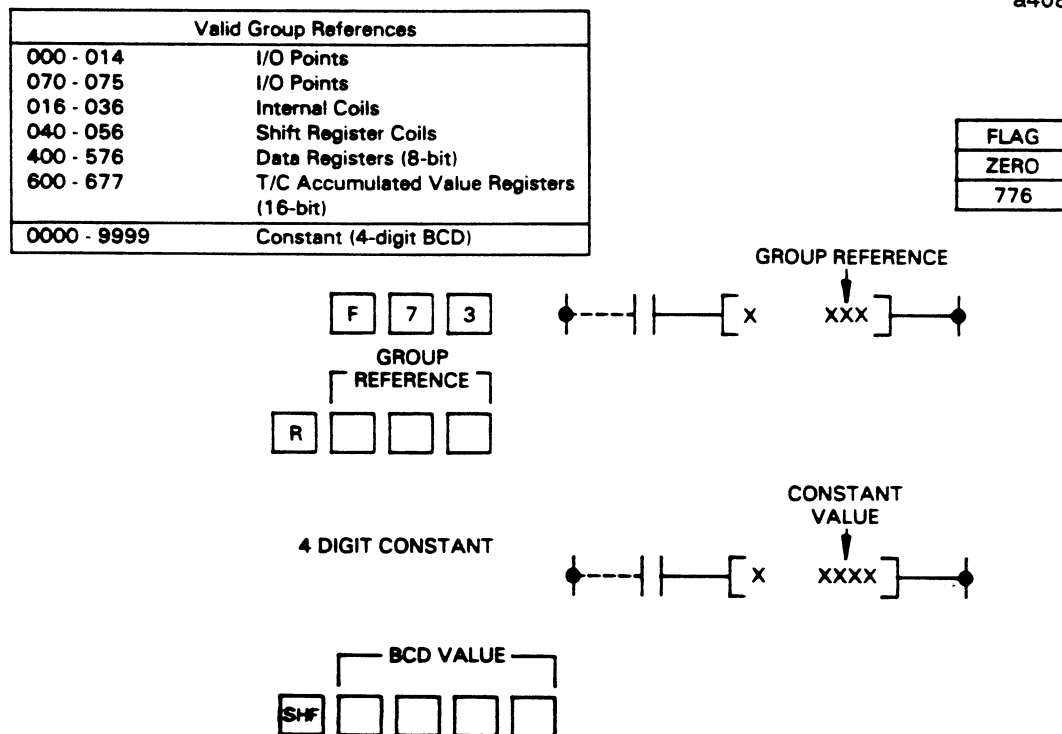
• Subtraction programming example

- Data is entered into the accumulator from Inputs 0040 - 0057.
- A constant value (3333) is subtracted from the contents of the accumulator.
- The answer is stored in the accumulator and is written to outputs 0150 - 0167.

Multiplication (X), BCD 4 Digit (F73)

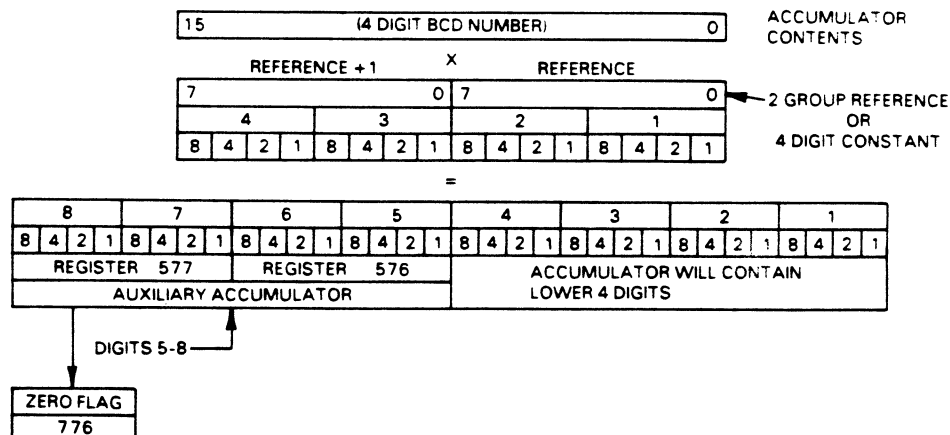
The contents of the accumulator are multiplied by the contents of a specified 2 group reference or a 4 digit BCD constant. The answer can be from 1 to 8 digits. The lower 4 digits of the answer are stored in the accumulator and digits 5-8 are stored in data registers 576 and 577, which are the auxiliary accumulator. If the answer is zero, the Zero flag (776) will turn ON.

a40870



• Multiplication sequence

a40871



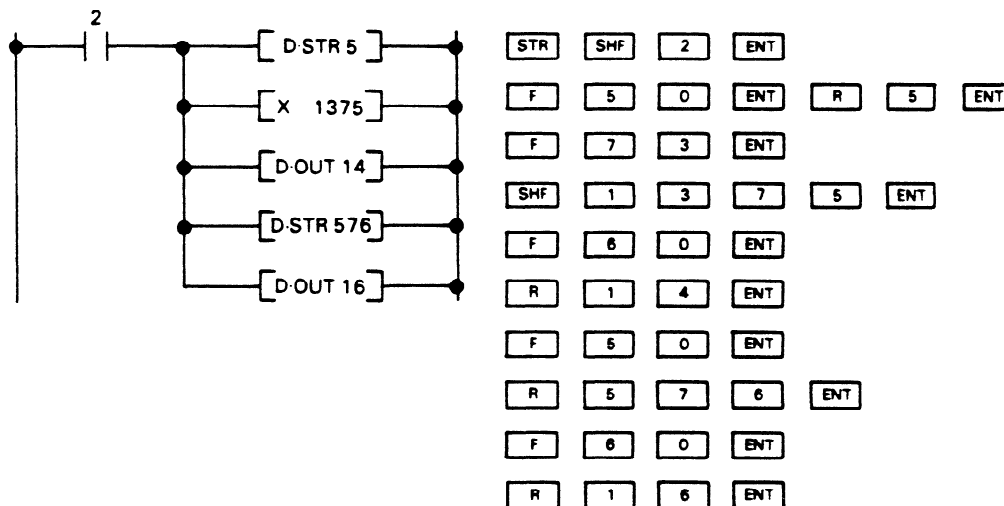
- The lower 4 digits (1, 2, 3, 4) of the answer will be stored in the accumulator. The upper 4 digits (5, 6, 7, 8) will be stored in the auxiliary accumulator (Data Registers 576 and 577).
- If the answer is 00000000, the Zero flag (776) is ON.

GEK-90842

• Multiplication programming example

Multiplying 4 digits by 4 digits with an answer containing 8 digits.

a40872



- A 4 digit BCD value is entered into the accumulator from Inputs 0050-0067.
- A constant value (1375) is entered as the multiplier.
- Assume that the answer is an 8 digit BCD number. The lower 4 digits are stored in the accumulator and transferred to Outputs 0140-0157. The upper 4 digits are stored in the auxiliary accumulator (Data Registers 576, 577) and transferred to Outputs 0160-0177.

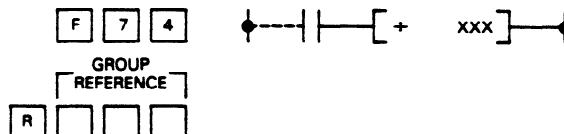
Division (+), BCD 4 Digit (F74)

The contents of the accumulator are divided by the contents of a specified 2 group reference or a 4 digit BCD constant. The 4 digits of the quotient are stored in the accumulator and the 4 digits of the remainder are stored in the auxiliary accumulator (Data Registers 576, 577). If either the dividend or divisor are zero, the Zero flag (776) will turn ON. If the divisor is zero, the Overflow flag (777) will also turn ON.

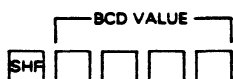
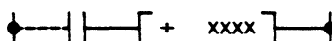
a40873

Valid Group References	
000 - 014	I/O Points
070 - 075	I/O Points
016 - 036	Internal Coils
040 - 056	Shift Register Coils
400 - 576	Data Registers (8-bit)
600 - 677	T/C Accumulated Value Registers
0000 - 9999	Constant (4-digit BCD)

FLAGS	
ZERO	OVERFLOW
776	777

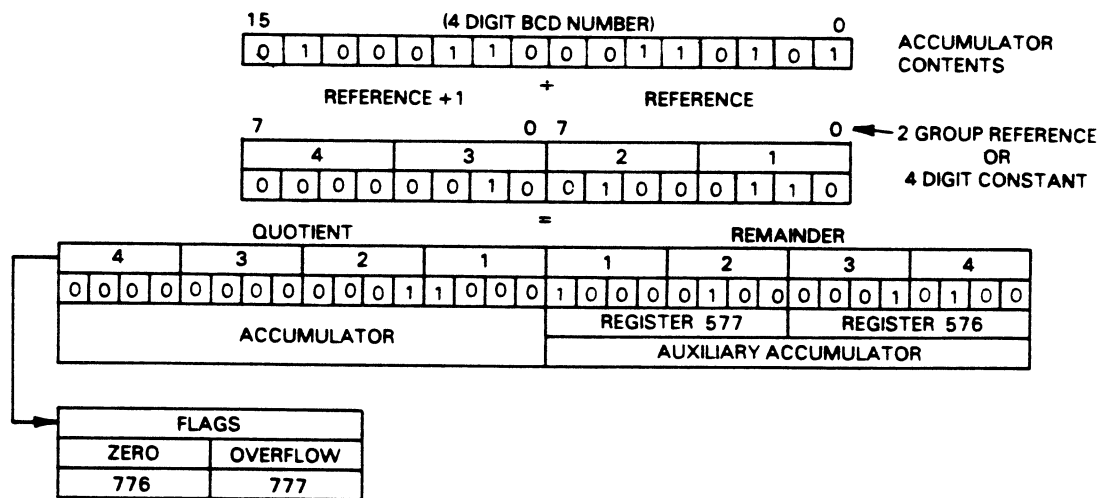


4 DIGIT CONSTANT VALUE



- Division sequence

a40874

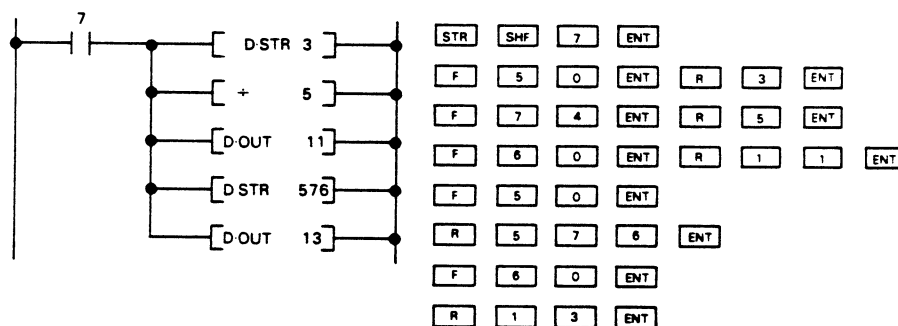


- The quotient (0 - 4 digits) will be stored in the accumulator as shown. the remainder (0 - 4 digits) will be stored in the auxiliary accumulator (Data Registers 577, 576) as shown.

- Division programming example

Division of a 4 digit number by another 4 digit number showing how the quotient and remainder can be written to Outputs.

a40875



- A 4 digit number is entered into the accumulator from Inputs 0030 - 0047.
- That number is divided by a number specified by Inputs 0050 - 0067.
- The 4 digits of the quotient are stored in the accumulator and are also written to Outputs 0110 - 0127.
- The 4 digits of the remainder are stored in the auxiliary accumulator (Data Registers 577, 576) and are written to Outputs 0130 - 0147.

GEK-90842

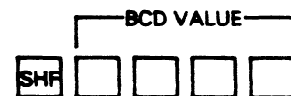
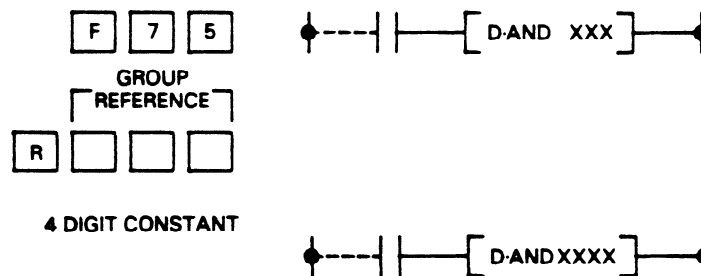
D•AND (DATA AND) (F75)

The contents of the accumulator are logically bit AND-ed with the contents of a specified 2 group reference or a 4 digit BCD constant. The result is stored in the accumulator. If the answer is 0000, the Zero flag (776) will turn ON.

a40876

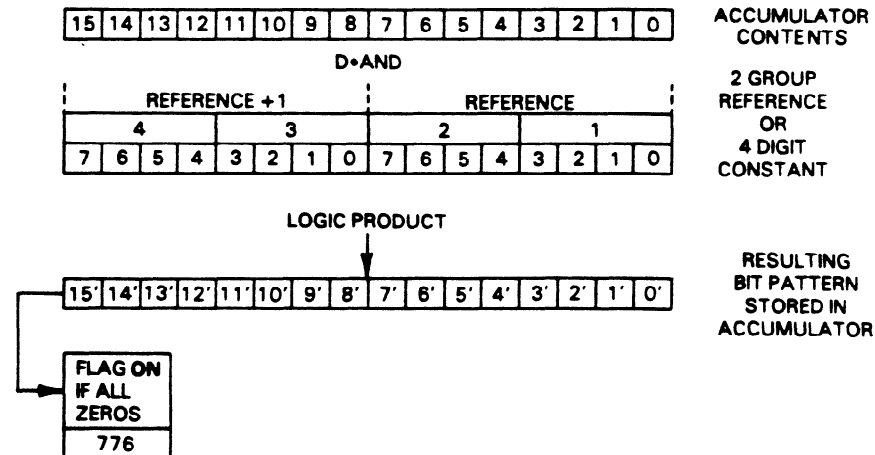
Valid Group References	
000 - 014	I/O Points
070 - 075	I/O Points
016 - 036	Internal Coils
040 - 056	Shift Register Coils
400 - 576	Data Registers (8-bit)
600 - 677	T/C Accumulated Value Registers (16-bit)
0000 - 9999	Constant (4-digit BCD)

FLAG
ZERO
776



D•AND Sequence

a40877

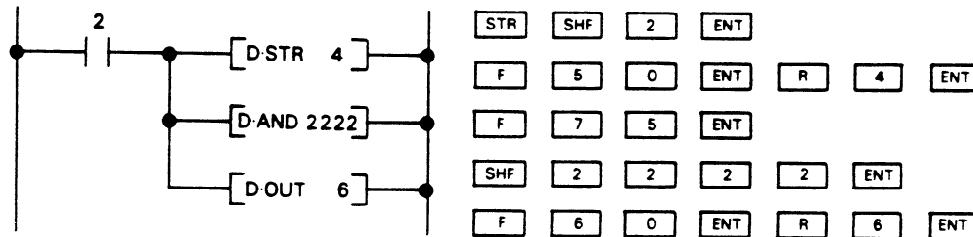


- Truth table for D•AND operation.
A = Accumulator contents
D = Contents of reference or constant

A	D	RESULT
0	0	0
0	1	0
1	0	0
1	1	1

• Programming example, D•AND

a40878



- Assume the data entered into the accumulator from Inputs 0040 - 0057 = 1234
- Is logically AND-ed with 2222
- The resulting answer stored in the accumulator would be 0220
- 0220 is written to Outputs 0060 - 0077
- Accumulator and reference contents for above operation are shown below

15	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0	0
7	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
15	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0

GEK-90842

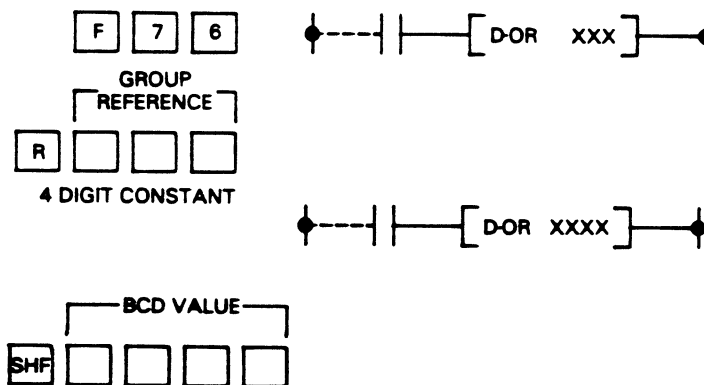
D•OR (DATA OR) (F76)

The contents of the accumulator are logically bit OR-ed with the contents of a specified 2 group reference or a 4 digit BCD constant. The result is stored in the accumulator. If the answer is 0000, the Zero flag (776) will turn ON.

Valid Group References	
000 - 014	I/O Points
070 - 075	I/O Points
016 - 036	Internal Coils
040 - 056	Shift Register Coils
400 - 576	Data Registers (8-bit)
600 - 677	T/C Accumulated Value Registers (16-bit)
0000 - 9999	Constant (4-digit BCD)

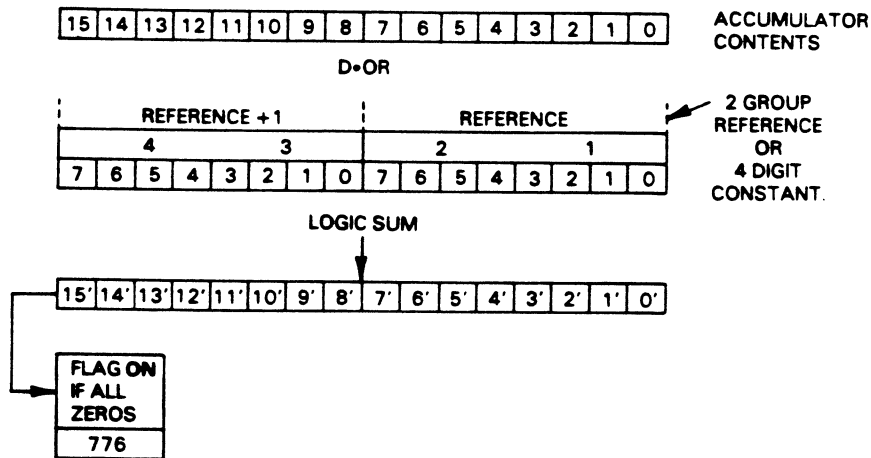
a40879

FLAG
ZERO
776



D•OR Sequence

a40880

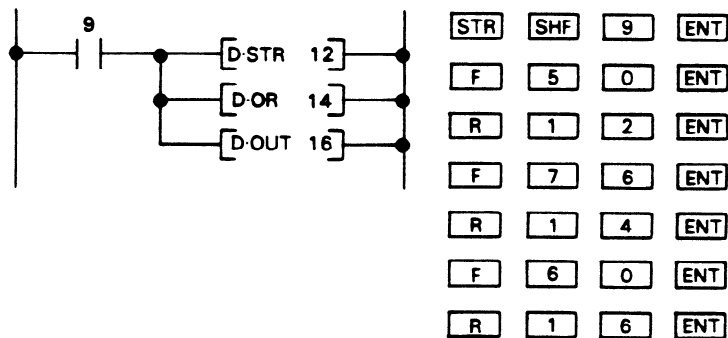


- Truth table for D•OR operation.
A = Accumulator contents
D = Contents of reference or constant

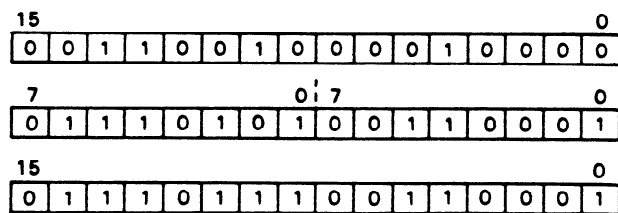
A	D	RESULT
0	0	0
0	1	1
1	0	1
1	1	1

• Programming example, D•OR

a40881



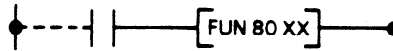
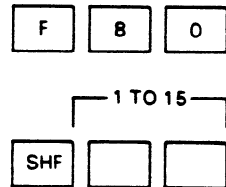
- Assume data entered into the accumulator from Inputs 0120 - 0137 = 3210
- Is logically OR-ed with data from Inputs 0140 - 0157 = 7531
- The resulting answer stored in the accumulator will be 7731
- 7731 is written to Outputs 0160 - 0177
- Accumulator and reference contents for above operation are shown below



GEK-90842

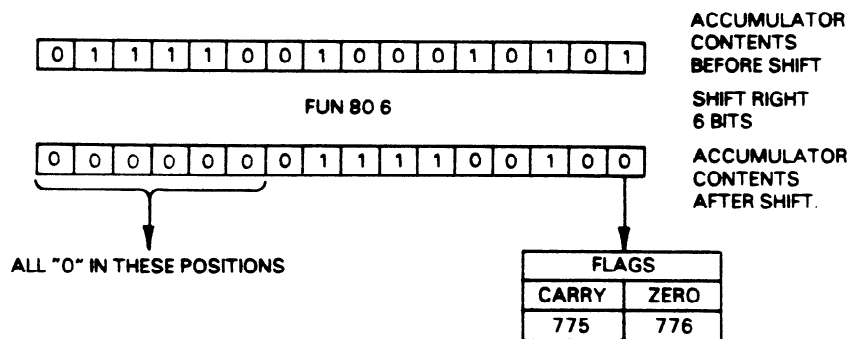
Shift Right (F80)

The contents of the accumulator are shifted to the right by the number of shifts (bits) specified by the instruction. The number of bits shifted can be from 1 to 15. The bit positions of bits shifted from the left are filled with zeros. If a '1' is shifted out of the accumulator as a result of the shift, the Carry flag (775) will turn on. If after the shift, the accumulator contains only zeros, the Zero flag (776) will turn on.



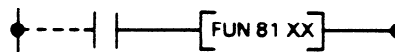
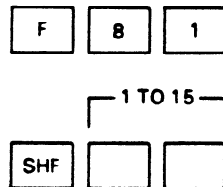
a40882

- FUN 80, Shift Right sequence (6 shifts in example)



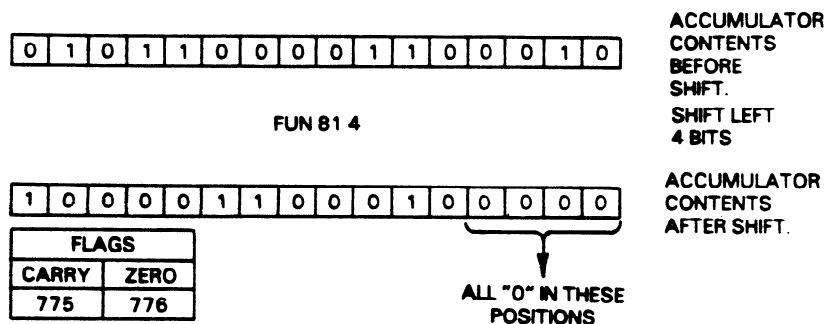
Shift Left (F81)

The contents of the accumulator are shifted to the left by the number of shifts (bits) specified by the instruction. The number of bits shifted can be from 1 to 15. The bit positions of bits shifted from the right are filled with zeros. If a '1' is shifted out of the accumulator as a result of the shift, the Carry flag (775) will turn on. If after the shift, the accumulator contains only zeros, the Zero flag (776) will turn on.



a40883

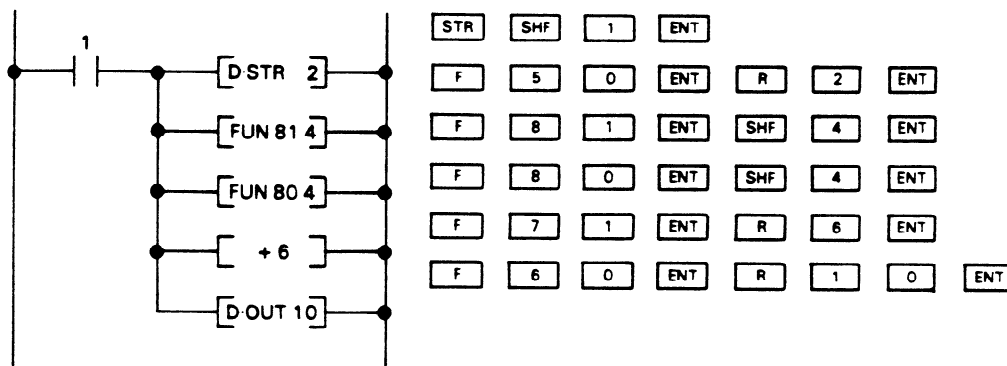
- FUN 81, Shift left sequence (4 shifts in example)



- Programming example using FUN 80 and FUN 81

This example shows a technique that can be used for deleting unnecessary data after it has been entered into the accumulator. In the example the fourth BCD digit will be deleted.

a40884



- Data entered into accumulator from Inputs 0020 - 0037.

•

- Shifted left, 4 bits.

•

- Shifted right, 4 bits.

•

- The fourth digit (5) has been deleted.

•

- The remaining value is added to the contents of Inputs 0060 - 0077. The result is stored in the accumulator and written to Outputs 0100 - 0117.

5	3	6	2
---	---	---	---

3	6	2	0
---	---	---	---

0	3	6	2
---	---	---	---

GEK-90842

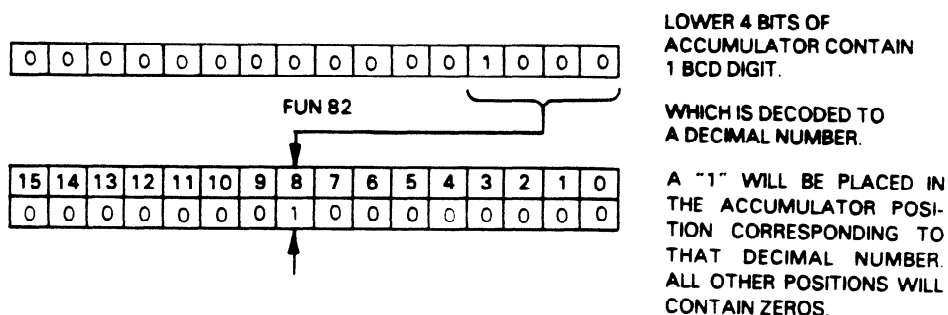
Decode (F82)

The lower 4 bits (least significant) of the accumulator are decoded to a decimal number from 0 to 15. A "1" is placed in the bit position in the accumulator that corresponds to the decoded decimal number. The upper 12 bits of the accumulator are disregarded for this operation.

a40885

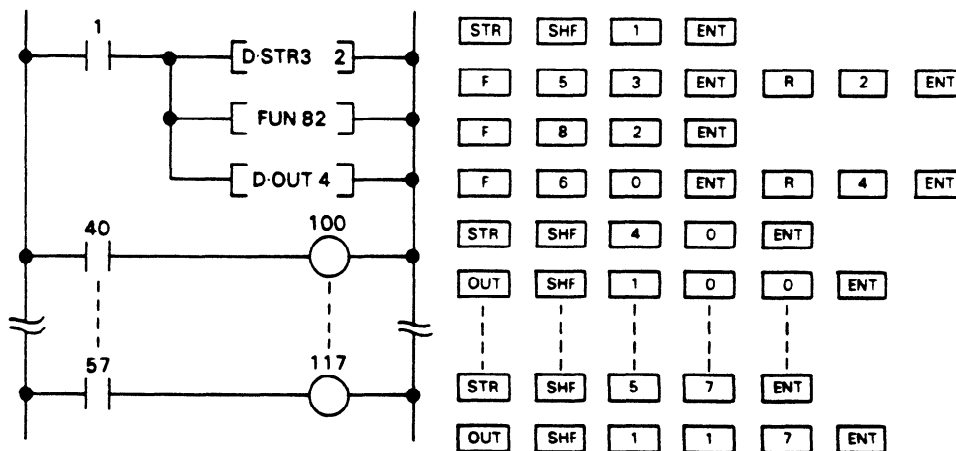


• FUN 82 Decode sequence



- If the BCD digit to be decoded is an 8, A "1" will be placed in the accumulator at that position as shown.
- Programming example, FUN 82

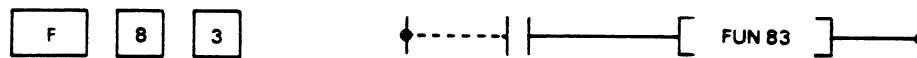
a40886



- A BCD digit (for example: 6) is loaded into the accumulator (lower 4 bits) from the lower 4 bits of the reference, Inputs 20, 21, 22, and 23. The bit pattern loaded in the accumulator would be 0 1 1 0.
- The BCD digit is decoded to a decimal number between 0 - 15 (in this example 6).
- That number is stored in the corresponding bit position in the accumulator.
- Outputs 0040 to 0057 correspond to the 16 bits of the accumulator. One of the outputs will be enabled when the accumulator position has a "1" placed in it. When the contact corresponding to the output reference is closed, a coil 100 - 117 will turn on.

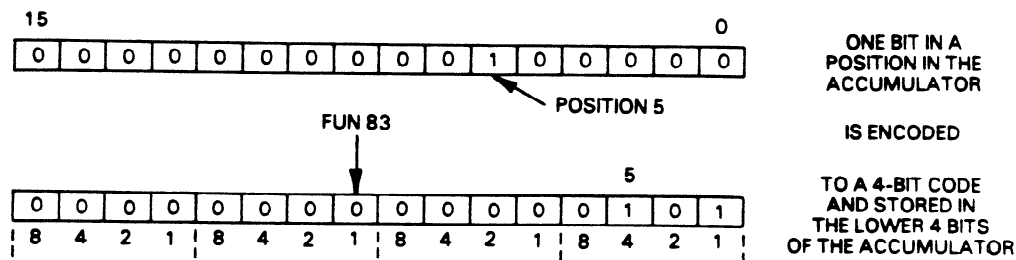
Encode (F83)

A bit in a position in the accumulator is encoded to a 4-bit code representing the position number (0-15) in the accumulator. The resulting 4-bit code is placed in the lower four bits of the accumulator.



a40887

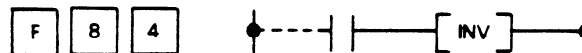
- FUN 83 Encode sequence (example)



- The upper 12 bits of the accumulator will contain zeros after the FUN 83 has been executed.

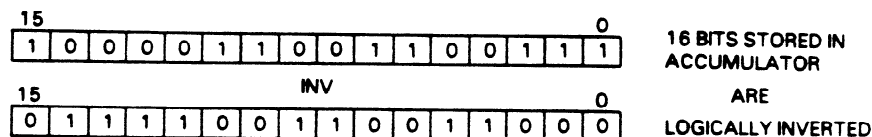
INV (F84)

The 16 bits stored in the accumulator are logically inverted; i.e., 1 to 0 and 0 to 1.

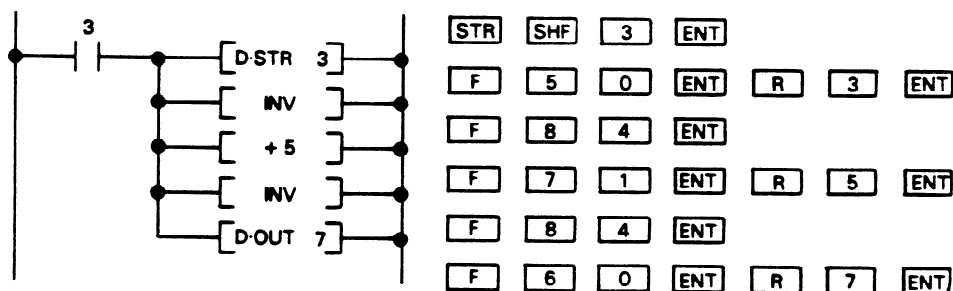


a40888

- Invert sequence



- Programming example, INV



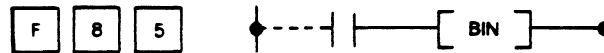
- Data entered into the accumulator from Inputs 0030 - 0047 is inverted.
- The inverted data is added to data from Inputs 0050 - 0067. The answer is stored in the accumulator, then inverted again.
- Data is written to Outputs 0070 - 0107

GEK-90842

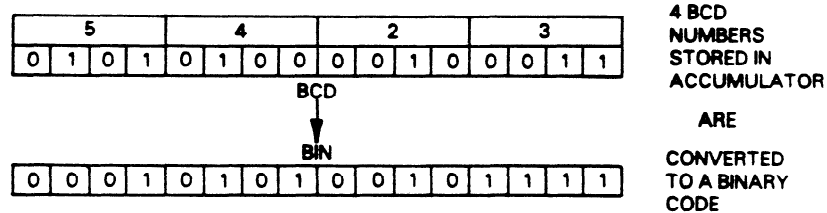
BIN (BCD to Binary) (F85)

The BCD value stored in the accumulator is converted to a binary code corresponding to that value.

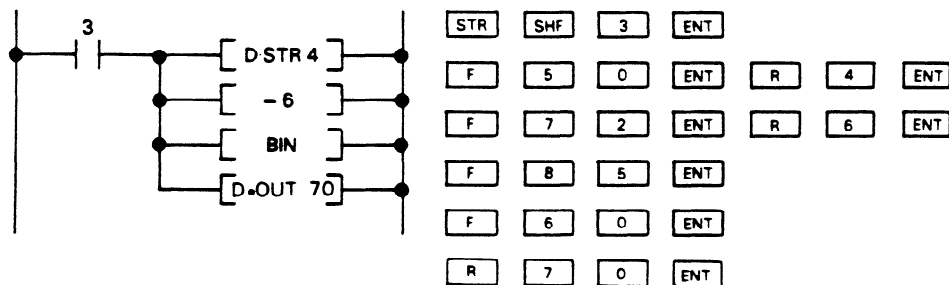
a40889



• Binary conversion sequence



• Programming example, BIN

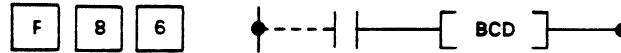


- Data is stored in the accumulator from Inputs 0040 - 0057.
- Data entered from Inputs 0060 - 0077 is subtracted from the value in the accumulator.
- The resulting answer is stored in the accumulator, where it is converted to a binary code.
- The binary code is written to Outputs 0700 - 0717.

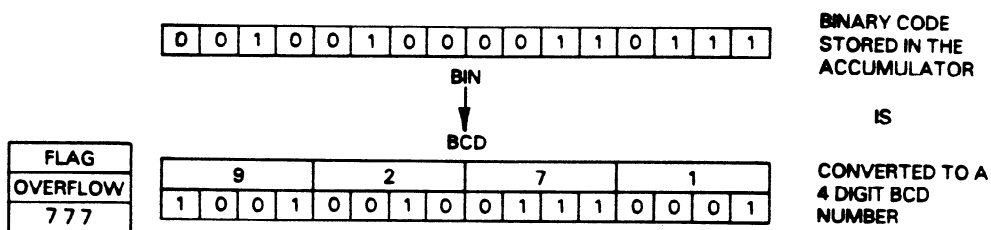
BCD (Binary to BCD) (F86)

A binary code stored in the accumulator is converted to a 4 digit BCD value. If the resulting BCD value is >9999, the overflow flag (777) will turn on. The flag coil can be used to turn on an output device as an indication of the overflow.

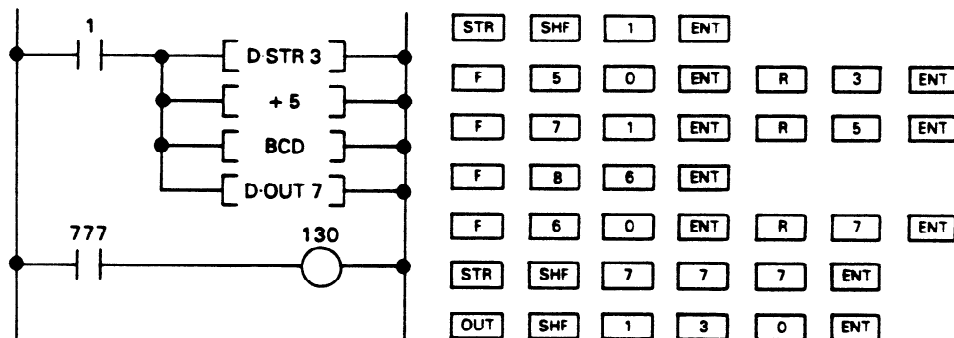
a40890



- BCD conversion sequence



- Programming example, BCD



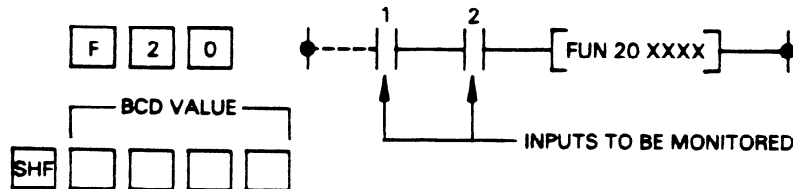
- Data is stored in the accumulator from Inputs 0030 - 0047.
- Data from Inputs 0050 - 0067 is added to the value in the accumulator.
- The resulting answer is stored in the accumulator, where it is converted to a 4 digit BCD number.
- The BCD number is written to Outputs 0070 - 0107.
- If an overflow occurs, the overflow Flag 777 will turn on Output 130.

GEK-90842

External Fault Diagnosis (F20)

This instruction provides a method of monitoring external devices for unwanted conditions such as switches that should not be on at the same time. When the instruction is programmed, a 4 digit BCD number is specified as a fault indication number. If the unwanted condition occurs, the 4 digit number will be displayed on the programmer data display. The 4 digit number can be a constant value or read from a group reference.

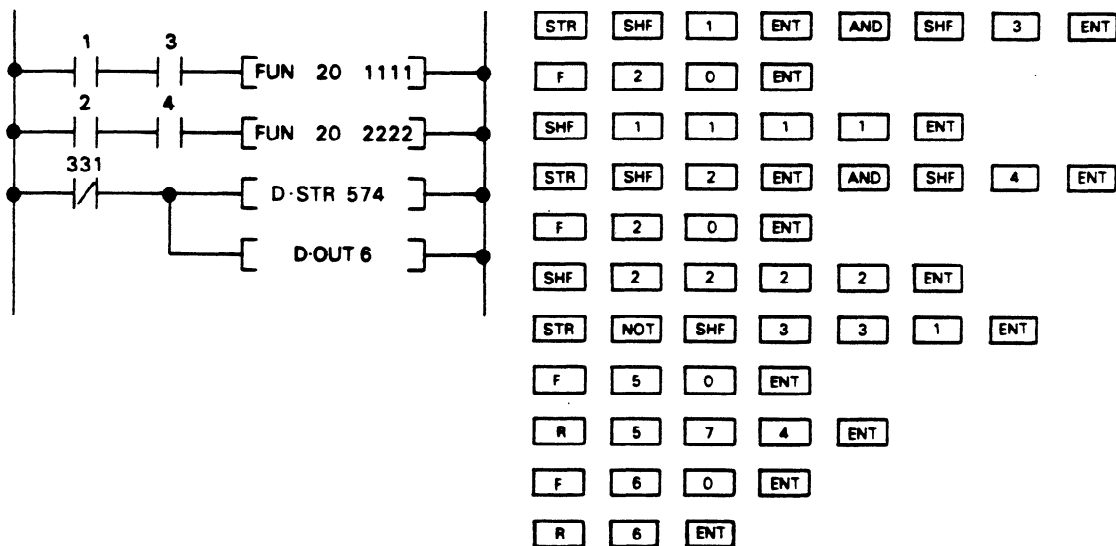
a40891



More than one FUN 20 instruction can be used in a program for detection of multiple fault conditions. Each condition to be monitored is assigned a different 4 digit number. If multiple faults occur simultaneously, the number displayed will be the one assigned by the last FUN 20 in the program. Data Registers 574 and 575 will contain the fault indicator number. When a fault number is displayed on the programmer, the fault number can be cleared from the display by depressing the CLR key. However, it is recommended that the actual fault be fixed before clearing the display.

• Programming example, FUN 20

a40892



- If devices connected to I1 and I3 are made at the same time, FUN 20 will be active, 1111 will be displayed on the data display on the programmer.
- If devices connected to I2 and I4 are made at the same time the data display will display 2222.
- The indicator number will be stored in 574 and 575 and that number will be written to Outputs 0060-0077.
- Fault indicator number display



- The physical fault should be cleared. Then clear the programmer display with the CLR key.

